PR1ME

Initial Documentation Release

IDR 3045
PRIME COMPUTER
FORTRAN
REFERENCE GUIDE FOR DBMS

First Printing July 1977

The information in this document is subject to change without notice and should not be construed as a commitment by Prime Computer Corporation. Prime Computer Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

ILLUSTRATIONS

TABLES

July 1977

FOREWORD

The FORTRAN Reference Manual for DBMS contains specifications of the
Data Description Language declaring a FORTRAN sub-schema (Section 2),
and the FORTRAN Data Manipulation Language (Section 3).

DOCUMENTATION EXCELLENCE

Prime is striving to maintain the highest documentation standards. To
achieve this goal, the Database documentation will be published in
three documentation releases as described in Section 1. This is the
Initial Documentation Release. Prime asks that each serious Database
user correspond his comments about this manual concerning technical
accuracy and additional information needed to implement the task of
Database Administrator.

Robert E. Dawes, Technical Writer
Technical Publications Department
Prime Computer Inc.
145 Pennsylvania Avenue,
Framingham, Ma. 01701

ACKNOWLEDGEMENT

Prime Computer, Inc. wishes to formally acknowledge the work of the CODASYL Programming Language Committee (PLC) and the Data Description Language Committee (DDLC). The Data Base Task Group (DBTG) of the PLC produced in April, 1971 a report containing the specifications of a standardized data base management facility consisting of a Data Description Language for describing a database, a Data Description Language for describing that part of the database known to a COBOL program, and a Data Manipulation Language for COBOL.

The Prime DBMS, portions of which are described in this manual, is based almost completely on the April 1971 DBTG specifications. Prime Computer is also participating in the ongoing work of CODASYL in the area of data base management through its membership on the DDLC and on the Data Base Language Task Group (DBLTG) of PLC.

SECTION 1

INTRODUCTION

ABOUT THIS MANUAL

This manual is oriented toward knowledgeable database application programmers. The reader is assumed to be acquainted with basic concepts of virtual memory operating systems and familiar with the benefits of database management principles.

Section 2 provides syntax rules and procedures for using the FORTRAN DATA DESCRIPTION LANGUAGE processor and Section 3 provides the description and procedurestekman etaoin using the DATA MANIPULATION LANGUAGE. Section 4 provides necessary diagnostic procedures for both processors.

DATABASE DOCUMENTATION

Database documentation (Figure 1-1) is provided for both the Database Administrator and the application programmer. The Database Administrator uses two manuals: 1) The Prime Database Administrator User's Guide and 2) Prime DBMS Schema Data Description Language (DDL) Reference Manuals.

The application programmer uses two manuals per application language: 1) the Prime FORTRAN Reference Manual for DBMS and the Companion Prime FORTRAN User's Guide, 2) The COBOL Reference Manual for DBMS and the Companion Prime COBOL User's Manual.
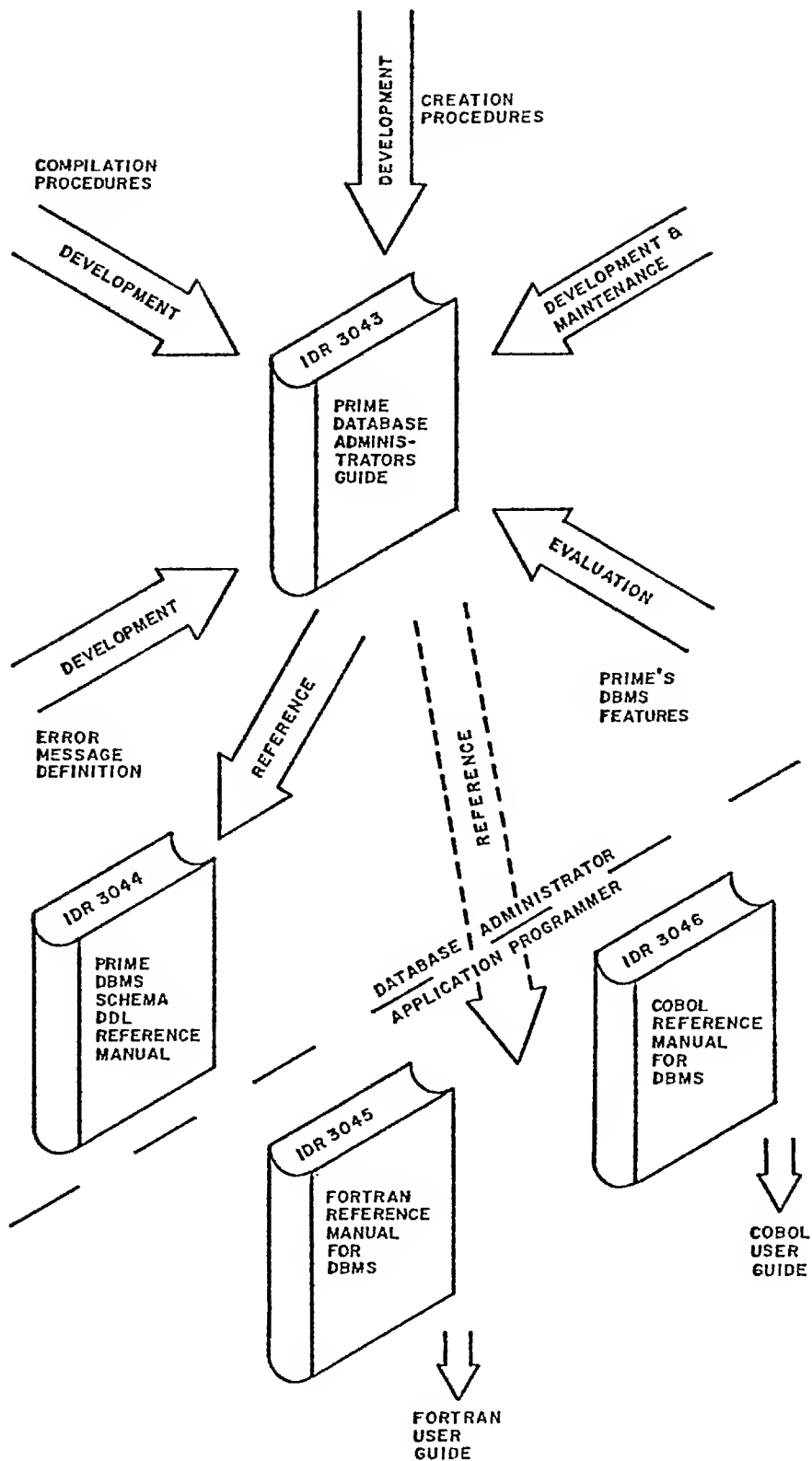
Figure 1-1.   PRIME DBMS DOCUMENTATION

Documentation Releases

Prime provides three documentation releases (see Figure 1-2) for every new product: The Initial Documentation Release (IDR), the Preliminary Documentation Release (PDR), and the Final Documentation Release (FDR).

The Initial Documentation Release (IDR) provides whatever information is available without regard to whether the information is grammatically correct, or properly formatted. Thus, the intent is to provide usable information when needed.

The Preliminary Documentation Release (PDR) is the second draft by the writer. It provides more complete and accurate information about the product, but does not represent the final product information. Customers having the IDR should request the PDR from his sales representative.

The Final Documentation Release (FDR) is the complete product description up to the stated software revision number. This release is edited, formatted and presented in Prime's highest professional standards. Users will be notified when this release is available and should contact the local sales representative for a copy.

OTHER RELATED PRIME DOCUMENTS

      PRIMOS FILE SYSTEM USER'S GUIDE   MAN2604

      PRIMOS INTERACTIVE USER'S GUIDE   MAN2602

      PRIMOS COMPUTER ROOM USER'S GUIDE   MAN2603

      PROGRAM DEVELOPMENT SOFTWARE USER'S GUIDE MAN1879

      FORTRAN IV USER'S GUIDE   MAN3057

      COBOL USER'S GUIDE   MAN2797

PRIME
INITIAL
DOCUMEN-
TATION
RELEASE
(IDR)

PROVIDES IMMEDIATE
INFORMATION

PRIME
PRELIMINARY
DOCUMEN-
TATION
RELEASE
(PDR)

EDITED & FORMATTED
INFORMATION

PRIME
FINAL
DOCUMEN-
TATION
RELEASE
(FDR)

BETTER
PRESENTATION
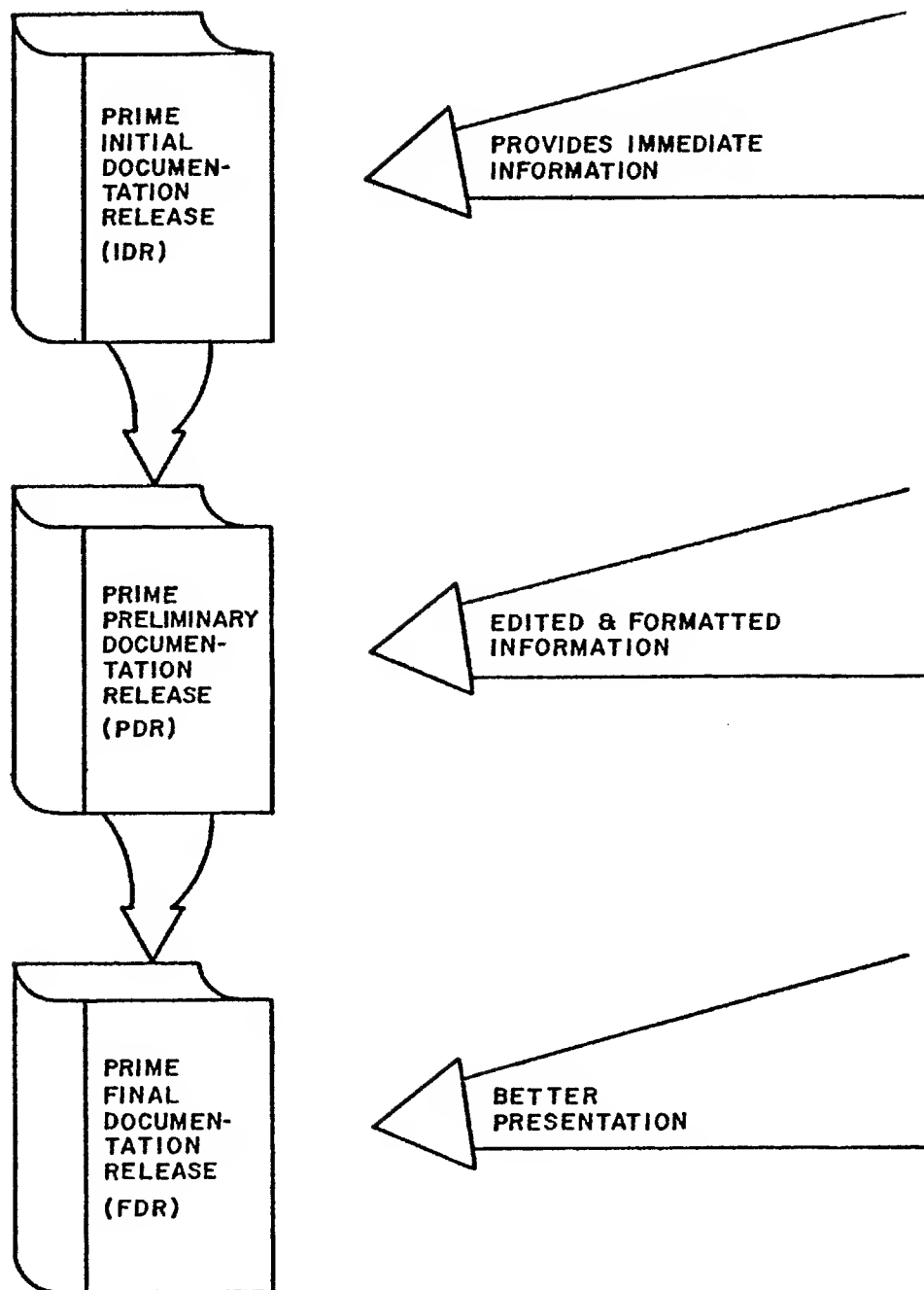
Figure 1-2.   DOCUMENTATION RELEASES

THE FORTRAN/DBMS INTERFACE

The FORTRAN interface to the DBMS includes two major processors and
their respective languages: the FORTRAN Subschema Data Definition
Language (DDL) Compiler and the FORTRAN Data Manipulation Language
(DML) Preprocessor.

The application programmer's "view" of a schema is written in the
FORTRAN Subschema DDL. The Subschema Compiler translates the DDL into
an internal, tabular form called the subschema table which is used by
the DML Preprocessor.

Commands for locating, retrieving, deleting, and modifying the contents
of a database are written in the FORTRAN DML. These commands are
interspersed with FORTRAN statements in the application program source
file and translated into FORTRAN declarations and statements by the
FORTRAN DML Preprocessor. The output of the Preprocessor is the source
input for the FORTRAN Compiler.

DATABASE DEVELOPMENT

The database development sequence of events is illustrated in figure
1-3.   Before performing any compilation, the user should examine this
flow to assure that the database is constructed in the proper sequence.
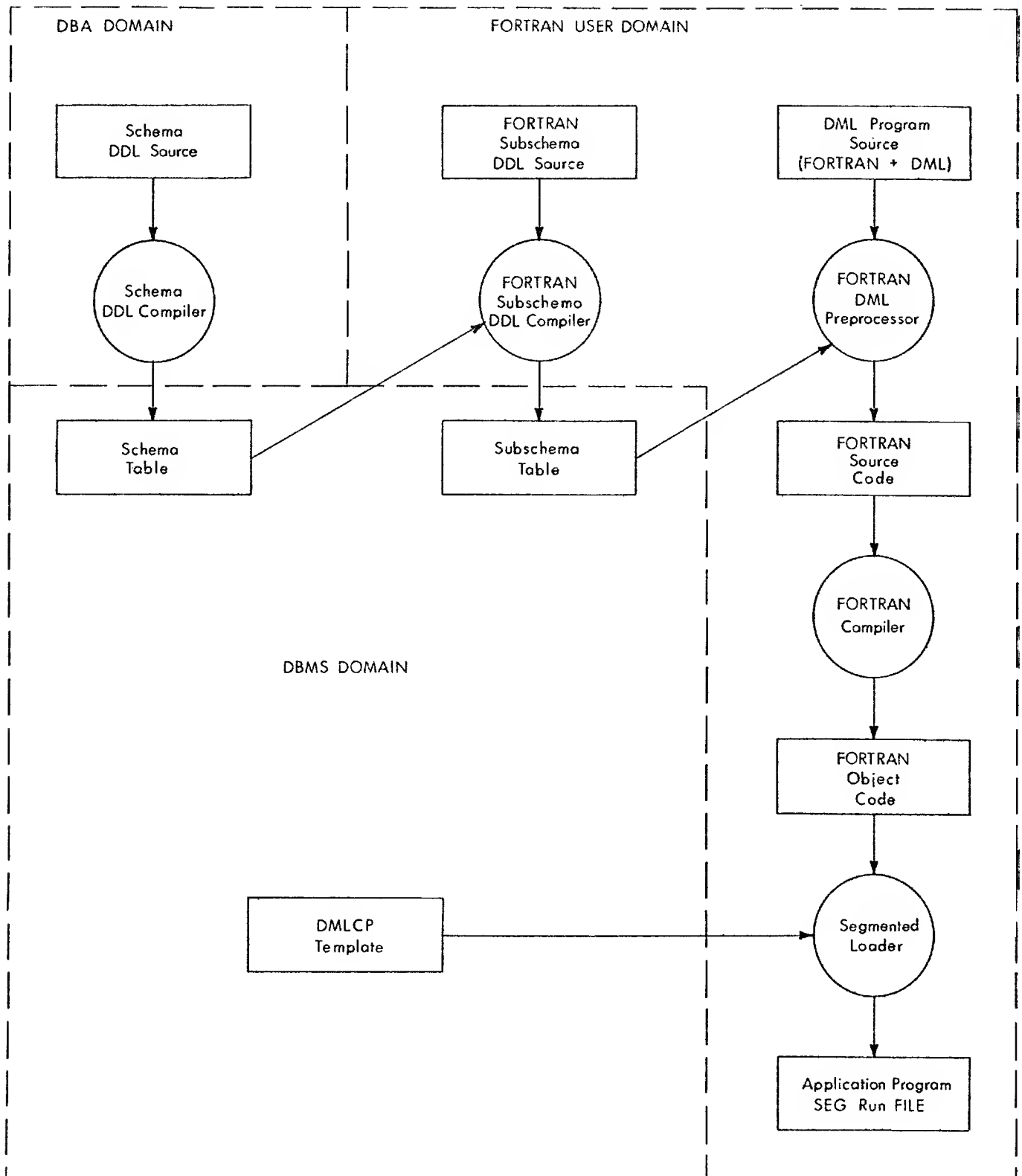
Figure 1-3   Database Development Sequence of Events

SECTION 2

FORTRAN DATA DESCRIPTION LANGUAGE COMPILER

PURPOSE

This section contains the complete specifications of the Data Description Language (DDL) for declaring a FORTRAN Subschema.

The specifications provide two divisions which serve to:

o   Identify the subschema (Subschema Identification Division)

o   Define (and optionally rename) the areas, records, items, and sets included in the subschema (Subschema Data Division)

The Subschema Identification Division must precede the Subschema Data Division and consists of a single entry. The Subschema Data Division consists of the following four types of entries:

DATA-BASE-DATA-NAME ENTRY (DBDN ENTRY)

AREA ENTRY

RECORD DESCRIPTION ENTRY

SET ENTRY

The DBDN ENTRY and the SET ENTRY are optional. A RECORD DESCRIPTION ENTRY is composed of one Record Control Entry and zero, one, or more Data Description Entries.

An entry may consist of one or more clauses. A clause specifies an attribute of an entry and is made up of phrases. All entries are terminated by periods. A semicolon is used to separate clauses.

For a RECORD DESCRIPTION ENTRY, a Complete Entry Skeleton is shown, representing the breakdown of the RECORD DESCRIPTION ENTRY into its component entries. For all other entries (including the component entries of a RECORD DESCRIPTION ENTRY) a Complete Entry consists of the following:

o   A narrative description of its function

o   Its general format

o   The syntax rules which apply

o   The general rules which apply

A general format is the arrangement of the elements which make up a clause. Inter-clause and intra-clause sequencing are governed by the sequence shown in the Complete Entry, except when otherwise specified in syntax rules. A syntax rule amplifies or restricts the usage of the elements within a general format. A general rule amplifies or restricts functions attributed to a general format or to its constituent elements.

The notation used in all formats and the rules which apply to all formats are:

o   The elements which make up a clause consist of upper-case words, lower-case words, special symbols, and special characters.

o   All underlined upper-case words are required when the format is used.

o   Upper-case words which are not underlined are optional words and need not be used.

o   lower-case words are generic terms which must be replaced by appropriate names or values.

o   The meaning of enclosing a portion of a general format in special symbols is:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix}$$ A source specification is not required in this position but it may contain either a, b, or c.

$$\begin{Bmatrix} a \\ b \\ c \end{Bmatrix}$$ A source specification in this position must must include exactly one of the components a, b, or c.

$$\begin{Vmatrix} a \\ b \\ c \end{Vmatrix}$$ A source specification in this position may consist of a list of one or more of a, b, or c. If more than one is specified, they must be separated.

An ellipsis (that is, '...') indicates repetition is allowed. The portion of the format which may be repeated is determined by the '[' or '{' which logically matches the ']' or '}' to the immediate left of the '...'.

USING THE COMPILER

The FORTRAN Subschema DDL Compiler (FSUBS) translates the FORTRAN Subschema Data Definition Language (DDL) into an internal tabular form called the Subschema table. FSUBS also produces an output listing which includes the DDL text, error messages, and a map of the User Work Area. The listing is found in the current UFD and is called "L<-name",

where "name" is the DDL input file name.

Invoking The Compiler

The FORTRAN Subschema DDL Compiler is invoked with the command:

FSUBS source-file [-VOL volume-name]

If volume-name is not specified for the output subschema table, the volume of the schema table is assumed.

Compiler Errors

There are three classes of errors that may be encountered by the compiler:

1. Fatal - cannot parse further.

2. Nonfatal - can continue parsing for further errors after end of this clause, but final subschema table will not be produced.

3. Warning - Syntax is ambiguous or incomplete and the compiler has made an arbitrary decision so that a final Subschema table may be produced.

Error messages are displayed both on the user terminal with the line in which the error occured, as well as in the output listing after the erroneous line.

User Work Area Map

A map of the User Work Area COMMON block is included in the output listing of the FORTRAN Subschema DDL Compiler. The map includes the name, dimensions, FORTRAN data type, and starting address in User Work Area for all items and data-base-data-names specified in the Subschema as well as the various DML run-time error and exception registers.

Schema/Subschema Linkage

If the output Subschema table is completed, it is automatically entered in the Subschema directory in the schema table of its parent schema, and can be accessed by the Database Administration Command Processor, the FORTRAN Data Manipulation Language Precompiler, and the DML Command Processor.

```
*****************
* SUBSCHEMA NAME *
*****************
```

IDENTIFICATION DIVISION

Function

To name a subschema and its associated schema.

General Format

SUBSCHEMA NAME IS subschema-name OF SCHEMA NAME schema-name

Syntax Rules

1. Subschema name must be unique among all subschema names for this
    schema name known to the DBMS.

2. The subschema name must be one to thirty alphanumeric characters
    or "-" or "*" with the first character being alphabetic.

General Rules

1. There must be one and only one SUBSCHEMA NAME clause and it must
    appear first in the subschema definition.

```
*****************
*  PRIVACY KEY   *
*****************
```

IDENTIFICATION DIVISION

Function

To specify the key for accessing the schema which includes the privacy
lock on its use for developing a subschema (that is, it includes a
PRIVACY LOCK FOR INCLUSION CLAUSE).

General Format

    PRIVACY KEY FOR INCLUDE IS literal-5

Syntax Rules

General Rules

   1. There may be only one PRIVACY KEY clause in the subschema.

DATA DIVISION

The SUBSCHEMA DATA DIVISION names the data-base-data-names, areas, records, and sets of the schema that are to be included in the sub-schema.

Structure of the Subschema Data Division

The Subschema Data Division consists of four types of entries:

DATA-BASE-DATA-NAME ENTRY

AREA ENTRY

RECORD DESCRIPTION ENTRY

SET ENTRY

The DATA-BASE-DATA-NAME ENTRY (DBDN) and the SET ENTRY may be omitted if not required. There must be at least one AREA ENTRY and RECORD DESCRIPTION ENTRY.

```
****************************
* DATA-BASE-DATA-NAME ENTRY *
****************************
```

DATA DIVISION

Function

To specify the data-base-data-names of the schema which will be used by the subschema. Optionally, to name the data-base-data-names in order to achieve conformity with the naming conventions of FORTRAN, or for convenience purposes.

General Format

INCLUDE DBDNS data-base-data-name-1 [= data-base-data-name-2]

[,data-base-data-name-3 [= data-base-data-name-4]]... .

Syntax Rules

1. Data-base-data-name-1 and data-base-data-name-3 must have been previously declared in the schema. Data-base-data-name-2 and data-base-data-name-4 must conform to FORTRAN naming rules. Data-base-data-name-2 is known as a synonym for data-base-data-name-1, to be used by the run-unit which invokes this subschema.

2. The collection of data-base-data-names for a subschema consists of:

    a. The synonyms defined in DBDN entries for a particular subschema.

    b. Those names declared in the schema which are known to the subschema and do not have synonyms in this subschema. The collection must not contain any duplicates.

3. Each name in the subschema collection of data-base-data-names must conform to the naming rules for FORTRAN.

General Rules

None.

```
**************
* AREA ENTRY *
**************
```

DATA DIVISION

Function

To enumerate the areas of the schema that are included in the subschema
and, by implication, to remove from view all other areas of the schema.

Optionally, to rename areas for convenience purposes.

General Format

INCLUDE AREAS area-name-1 [=area name-2] [,area-name-3 [=area-name-4]] ... .

Syntax Rules

1. Area-name-1 and area-name-3 must refer to areas defined in the
   schema.

2. Area-name-2 and area-name-4 are synonyms for area-name-1 and
   area-name-3, respectively. The synonyms are to be used by a
   run-unit which invokes this subschema.

3. The collection of area-names known to the subschema consists of:

   a. The synonyms defined in AREA entries.

   b. The area-names declared in AREA entries which do not have
      synonyms. The collection must not contain any duplicates.

```
***************************
* RECORD DESCRIPTION ENTRY *
***************************
```

DATA DIVISION

Function

To enumerate and define the records and subordinate data items within records of the schema that are to be included in the subschema.

By implication, to remove from view all other records and data items within records of the schema.

Complete Record Description Entry Skeleton

Record Control Entry

     [;Data Description Entry]... .

Complete Record Control Entry

     INCLUDE RECORD record-name-1[=record-name-2] [OVERLAYED WITH RECORD record-name-3]

Complete Data Description Entry

$$\begin{bmatrix} \text{INTEGER*2} \\ \text{INTEGER*4} \\ \text{REAL*4} \\ \text{REAL*8} \\ \text{LOGICAL} \\ \\ \text{DBK} \\ \\ \text{CHARACTER} \end{bmatrix}$$ data-name-1 [=data-name-2][dimension specification]

Syntax Rules for Complete Record Description Entry

1. A Record Control Entry is required as the first element in a subschema record description; it may be followed by zero or more Data Description Entries.

General Rules for Complete Record Description Entry

1. A Data Description Entry is used to describe an elementary data item of the associated schema, whether or not it is included in one or more repeating groups.

```
***************
* RECORD NAME *
***************
```

DATA DIVISION

Function

To define records which are to be included in the subschema.

Optionally, to rename records for convenience purposes.

INCLUDE RECORD record-name-1 [=record-name-2] [OVERLAYED WITH RECORD record-name-3]

Syntax Rules

1. Record-name-1 must refer to a record declared in the schema.

2. Record-name-2 is synonymous for record-name-1 to be used by a run-unit which invokes this subschema.

3. The collection of record-names known to the subschema consists of:

   a. The synonyms defined in the Record Control Entries.

   b. The record-names declared in the Record Control Entries which do not have synonyms. The collection must not contain any duplicates.

4. All record-names appearing in Record Control Entries must be unique.

5. Record-name-3 is the subschema name of a record previously defined in this subschema. The record defined by this statement will be overlayed with the record identified by record-name-3 in the User Work Area COMMON block, both starting at the same address.

General Rules

1. If OVERLAYED is specified, the record defined by record-name-1 must be less than or equal in length to the record identified by record-name-3 in the User Work Area COMMON block.

```
**********************
* TYPE SPECIFICATION *
**********************
```

DATA DIVISION

Function

To specify the characteristics of a data item in the subschema.

By implication, to determine conversion rules between schema and subschema data items.

General Format

```
┌                ┐
│  INTEGER*2     │
│                │
│  INTEGER*4     │
│                │
│  REAL*4        │
│                │
│  REAL*8        │
│                │
│  LOGICAL       │
│                │
│  DBK           │
│                │
│  CHARACTER     │
└                ┘
```

Syntax Rules

   1. If type CHARACTER is specified, the first dimension given
      multiplied by 2 is the number of characters in the string. The
      string is packed two characters to a word in a FORTRAN integer
      array in the User Work Area COMMON block.

General Rules

   1. The valid conversions are listed in Table 2-1. Any conversion
      not listed is not valid. Invalid conversions cause an Error
      Status Condition when attempted by the DML command processor.

Table 2-1. Schema/FORTRAN Subschema Type Conversions

2. Conversions permitted from FORTRAN subschema to   schema   or   from schema to FORTRAN subschema.

from   to

Type	Type	Description of Conversion
DBK	DBK	OK
CHARACTER	CHARACTER	Truncated on right or blank-filled on right as necessary.
CHARACTER	INTEGER*2	Ignores all blanks in the string.
CHARACTER	INTEGER*4	Ignores all blanks in the string.
CHARACTER	REAL*4	Uses FORTRAN DECODE with FORMAT (E14.6); string must be 14 characters.
CHARACTER	REAL*8	Uses FORTRAN DECODE with FORMAT (E24.6); string must be 24 characters.
INTEGER*2	CHARACTER	Uses FORTRAN ENCODE with FORMAT (I6, 999X); string must be at least 6 characters.
INTEGER*4	CHARACTER	Uses FORTRAN ENCODE with FORMAT (I12, 999X); string must be at least 12 characters.
REAL*4	CHARACTER	Uses FORTRAN ENCODE with FORMAT (1PE14.6, 999X); string must be at least 14 characters.
REAL*8	CHARACTER	Uses FORTRAN ENCODE with FORMAT (1PE24.14, 999X); string must be at least 24 characters.
INTEGER*2	INTEGER*2	All conversions are valid; uses FORTRAN conversions.
INTEGER*4	INTEGER*4	
REAL*4	REAL*4	
REAL*8	REAL*8	

3. Conversions permitted from FORTRAN subschema to schema only.

Subschema   Schema

Type	Type	Description of Conversion
CHARACTER	CODE	See General Rule 9.
CHARACTER	DATE	See General Rule 6.
CHARACTER	TIME	See General Rule 8.
INTEGER*2	BIT	Appropriate number of words moved without conversion; zero-filled on right.
INTEGER*2	CODE	See General Rule 10.
INTEGER*2	DATE	Same as INTEGER*2 to INTEGER*4.
INTEGER*2	TIME	Same as INTEGER*2 to INTEGER*4.
INTEGER*4	BIT	Appropriate number of words moved without converstion; zero-filled on right.
INTEGER*4	DATE	Moved without conversion.
INTEGER*4	TIME	Moved without conversion.
LOGICAL	BIT	Each logical word is placed in a bit; .TRUE. becomes 1; .FALSE. becomes 0.

4. Conversions permitted from schema to FORTRAN subschema only.

Schema   Subschema

Type	Type	Description of Conversion
BIT	INTEGER*2	Moves the appropriate number of words without conversion; zero-filled on the right.
BIT	INTEGER*4	Moves the appropriate number of words without conversion; zero-filled on the right.
BIT	LOGICAL	Each bit is moved to a word; 1 becomes .TRUE.; 0 becomes .FALSE..
CODE	CHARACTER	See General Rule 9.
CODE	INTEGER*2	See General Rule 10.
DATE	CHARACTER	See General Rule 5.
DATE	INTEGER*2	Same as INTEGER*4 to INTEGER*2.

DATE   INTEGER*4   Moved without conversion.

TIME   CHARACTER   See General Rule 7.

TIME   INTEGER*2   Same as INTEGER*4 to INTEGER*2.

TIME   INTEGER*4   Moved without conversion.

5. For conversions from DATE to CHARACTER, the string must be at
    least 12 characters long. DATE is an INTEGER*4 number
    representing the number of days since the beginning of the
    Gregorian calendar (Oct. 15, 1582). The format of the string
    will be `MON DD, yyyy' where 'MON' represents the 3-character
    abbreviation of the month, 'DD' represents the day, and 'yyyy'
    represents the 4-digit year. The date is left-justified and
    blank-filled on the right. An example date to character
    conversion is 'Jun 21, 1977'.

6. For conversions from CHARACTER to DATE, the following formats are
    permitted:

                    MON DD, YY

                    MM/DD/YY

                    MM-DD-YY

                    MMDDY

    Where 'MON' represents the 3-letter abbreviation of the month,
    'MM' represents a 1-digit or 2-digit month, 'DD' represents a
    1-digit or 2-digit day. 'YY' represents a 2-digit or 4-digit
    year, and 'Y' represents a 1-digit year. If the format is
    'MMDDY', the base decade (for example, 1970) is added to the
    year. If the format is 'MM/DD/YY' or 'MM-DD-YY' and the year is
    less than 100, then the base century (for example, 1900) is added
    to the year.

7. For conversions from TIME to CHARACTER, the string must be at
    least 8 characters long. TIME is an INTEGER*4 number
    representing the number of seconds since midnight. The format of
    the STRING is 'HH:MM:SS' where 'HH' represents the 2-digit hours,
    'MM' represents the 2-digit minutes, and 'SS' represents the
    2-digit seconds.

8. For conversions from CHARACTER to TIME, the string may have one of the following formats:

    HH

    HH:MM

    HH:MM:SS

   Where HH represents a 1 or 2-digit number for hours, 'MM' represents a 1 or 2-digit number for minutes, and 'SS' represents a 1 or 2-digit number for seconds.

9. For conversion from CODE to CHARACTER, the character string (1-30 characters) representing the appropriate encoding as specified in the CODE USING clause in the schema type declaration is put in the User Work Area COMMON block with trailing blanks. For conversion from CHARACTER to CODE, the string in the User Work Area must match one of the literals in the CODE USING clause, the internal pointer for which is stored in the record occurrence.

10. Conversions between CODE and INTEGER are the same as between CODE and CHARACTER except that instead of the literal character string, the number representing the order of the literal in the CODE USING list is found in the User Work Area COMMON block.

```
*************
* DATA NAME *
*************
```

DATA DIVISION

Function

To select data items of interest (whether or not they are included in
one or more repeating groups) from a record defined in the schema,
thereby implicitly removing from view all unnamed items of the record.

To create a FORTRAN non-hierarchical record description by including
data items from the schema and specifying them in terms of FORTRAN
variables and arrays.

Optionally, to rename data items in order to conform to the FORTRAN
naming conventions, or for convenience purposes.

General Format

data-name-1 [=data-name-2]

Syntax Rules

1. Data-name-1 must refer to a data item declared for the record in
    the schema whether or not it is included in a repeating group
    (i.e., data-name-1 may refer to a vector but not to a repeating
    group).

2. Data-name-2 must conform to FORTRAN naming rules. Data-name-2 is
    a synonym to be used by a run-unit which invokes this subschema.

3. The collection of data-names known to the subschema consists of:

    a. The synonyms defined in the Data Entries.

    b. The data-names declared in Data Entries which do not have
        synonyms. The collection must not contain any duplicates.

4. Each name in the subschema collection of data-names must conform
    to the naming rules for FORTRAN.

```
**************************
* DIMENSION SPECIFICATION *
**************************
```

DATA DIVISION

Function

To define data items which are vectors and/or participate in repeating groups. To dimension a FORTRAN variable, INTEGER or LOGICAL, so that it will be compatible with the length of a CHARACTER or BIT STRING data item in the schema.

General Format

      (integer-1 [,integer-2] [,integer-3] [,integer-4] [,integer-5]
           [,integer-6] [,integer-7])

Syntax Rules

1. Each integer specifies the maximum index value for each dimension in the normal FORTRAN manner (left-most changing "fastest").

2. Any data item of the schema included in one or more repeating groups must be declared as an array of the subschema, with number of dimensions inclusively between one and the number of nested repeating groups containing that data item, plus one dimension if the item is a vector and another if the item is type string. However, the number of dimensions cannot exceed the limit allowed by FORTRAN--currently three. The correspondence between the occurrences of the data item in the database and the array elements of the subschema is determined as explained by General Rules 2-5.

3. If the data item in the subschema is declared to be type CHARACTER, then a minimum of two dimensions must be specified, if the item is a vector and/or participates in a repeating group. If the item is not a vector and does not participate in a repeating group, a dimension need be specified only if CHARACTER strings of more than 2 characters in length are desired.

General Rules

1. For a data item which is declared as a vector and/or participates in a repeating group in the schema, the dimension specification restricts the run-unit's "view" of the occurrences of the item in a record occurrence to the number of elements available in the FORTRAN array.

2. If a one dimensional array is declared (or two dimensional if type CHARACTER), the occurrences are stored in the array arranged by higher to lower repeating group occurrences (with "higher" meaning the largest level number). That is, all occurrences of a data item at the highest repeating group level and corresponding to a set of fixed indices for the lower repeating groups are stored in consecutive array elements.

3. For each additional dimension, the n'th dimension from the right corresponds to the repeating group at level n below this item.

4. The number of dimensions specified - minus one if the item is type CHARACTER - is m and the number of repeating group levels to an item - plus one if the item vector is n. Then, if m<n, the elements available for storing levels m and all higher levels, including the item itself if it is a vector, are those specified by the left-most dimension (or second dimension if type CHARACTER).

5. Within each repeating group level, the occurrences in the array are indexed in the same order as in the database.

```
*************
* SET ENTRY *
*************
```

DATA DIVISION

Function

To enumerate and define the sets of the schema that are to be included in the subschema and by implication, to remove from view all other sets of the schema.

Optionally, to rename sets for convenience purposes.

General Format

INCLUDE SETS set-name-1 [=set-name-2] [,set-name-3 [-set-name-4]]...

Syntax Rules

1. Set-name-1 and set-name-3 must refer to sets defined in the schema.

2. Set-name-2 and set-name-4 are synonyms for set-name-1 and set-name-3, respectively, to be used by a run-unit which invokes this subschema.

3. The collection of set-names known to the subschema consists of:

    a. The synonyms defined in the SET Entries.

    b. The set-names declared in the SET Entries which do not have synonyms.

        The collection must not contain any duplicates.

General Rules

1. A RECORD DESCRIPTION ENTRY for the owner record of each set included in a subschema must also be included in that subschema.

SECTION 3

FORTRAN DATA MANIPULATION LANGUAGE PROCESSOR (FDML)

OVERVIEW

The purpose of the FORTRAN Data Manipulation Language (FDML) is to support access to the Data Manipulation Command Processor (DMLCP) using FORTRAN as a host language. The FDML language consists of English-like statements and is based on the April 71 report from CODASYL. These statements are translated by the FDML preprocessor into FORTRAN declarations, FORTRAN call statements, and other FORTRAN procedure statements.

Before a FDML program can be written, a FORTRAN SUBSCHEMA must be written and compiled by the FORTRAN SUBSCHEMA compiler (FSUBS) described in Section 2 of this manual. The FORTRAN SUBSCHEMA contains the names and declarations of the areas, sets, items, records and data-base-data-names that the FDML application program can reference.

HOW TO USE THE FDML PREPROCESSOR

Programming Tips

Once the subschema has been provided, the following programming tips should be considered when writing application programs.

1. All FORTRAN program units which contain FDML commands must contain either a SUBSCHEMA or DBMS SUBPROGRAM statement.

2. Only one subschema of one schema can be invoked at any one time in a DBMS application program.

3. The application program and subprograms which contain DML commands must be programmed in one language, e.g., all FORTRAN or all COBOL.

4. The FORTRAN DML application program must reference a FORTRAN SUBSCHEMA.

5. The FORTRAN application program should avoid using the COMMON blockname and the User Work Area register names which are output by the FORTRAN SUBSCHEMA compiler.

6. The procedure section of the FORTRAN application program may freely intermix FORTRAN statements and FDML statements; however, FDML statements and FORTRAN statements may not be mixed on the same line or on a continuation line.

7. Only one INVOKE statement should be executed during the duration of a run-unit. The execution of more than one INVOKE has undefined results. The failure to execute an INVOKE statement will cause the FDML statements to be ignored.

8. All FDML statements must be preceded by a '#' to identify the statement to the FDML preprocessor.

9. All FDML programs must terminate execution with an EXIT DBMS statement.

FDML ORGANIZATION

Two Classes of FDML Statements

There are two general classes of statements in the FDML, declaration statements and executable commands.

The declaration statements allow the FDML program to declare the SCHEMA and the SUBSCHEMA required for the application program. It also inserts the definition of a FORTRAN COMMON block which serves as the User Work Area (UWA) for the application program. The COMMON block contains the items and their types declared by the subschema and the DMLCP registers.

The FDML executable commands specify the individual DML processes required by the FDML application program. Currently the following FDML capabilities exist:

Run-Unit Oriented Commands

Run-Unit Oriented Commands are used by the run-unit to initiate and terminate communication with the DMLCP. They also allow the user to clear error statuses and declare Privacy locks. They include: INVOKE, EXIT DBMS, ABORT DBMS, CLEAR ERROR and PRIVACY KEY.

Area Oriented Commands

Area Oriented Commands obtain and relinquish control of the DBMS area files. They also allow the user to specify the mode in which these files are to be used. They include: OPEN and CLOSE.

Record Oriented Commands

Record Oriented Commands are used to create, manipulate, remove and obtain records from the database. These commands include: STORE, DELETE, GET, MODIFY, FIND and FETCH.

Set Oriented Commands

Set Oriented Commands manipulate set membership.   They   include:
INSERT, REMOVE and MODIFY.

Supporting Commands

Supporting   Commands   allow   the   user to manipulate currency statuses.
They include: SUPPRESS, MOVE and CLEAR.

Concurrency Functions

Concurrency Functions allow a run-unit to access and update a database
along with concurrent run-units.   They insure that the view of the
database is consistent, guarantee its integrity and provide   for
rollback and recovery facilities.   These commands include: START
TRANSACTION, END TRANSACTION and ABORT TRANSACTION.

FDML SYNTAX COMPONENTS AND NOTATION

This section describes the requirements   for   the   FORTRAN   Data
Manipulation Language (FDML) statements and syntax notation used in
programming FORTRAN Application Programs to run on Prime's DBMS system.
The first subsection describes the syntax components which may   appear
in an FDML statement.   The next subsection describes the syntax
notation which will be used in defining the permissible syntax within
the statements.

FDML Statement Format Rules

The FDML statements are in an English-like format but they have some of
the characteristics of a FORTRAN statement. They must start after
column 6 and end before column 73. A "C" in column 1 designates a
comment card.   A statement label may be inserted in columns 1-5, and a
statement may be continued by placing a non-blank character in column
6.   There is no limit to the number of continuation statements that may
be used.   In addition to these rules, all FDML statements and
declarations must start with a "#" character after column 6 to be
recognized by the FDML preprocessor.

Character Set

The FDML statements are composed of tokens which are themselves
composed from the character sets shown in Table 3-1.

Terminating FDML Statements

FDML statements are terminated by a period;   although   the   period   is
optional,   it should be used to improve error checking and to eliminate
usual error messages.

*

Table 3-1. FDML STATEMENTS CHARACTER SET

CHARACTER GROUP	MEMBERS
Letters	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
DIGITS	1 2 3 4 5 6 7 8 9 0
Special Characters	blank (space) - (minus sign or dash) , (comma) ; (semicolon) . (period) ' (quotation mark)
Reserved for future use	= (equal sign) > (greater than sign) < (less than sign) * (Asterisk) $ (Dollar sign)

Delimiting Characters

The following FDML characters serve to delimit syntax components: 1) space, 2) comma, 3) semicolon and 4) period. Consecutive spaces are treated as a single space. It should be noted that although a space is a delimiting character for an FDML statement, it is not a delimiting character in a FORTRAN statement. The end of a line also serves as a delimiting character so that a syntax component may not cross a line boundary.

Generic Terms

The following is a list of generic terms for the FDML Language and the rules for their function.

Schema-name

Used to identify that schema the run-unit is to access. It must identify a valid schema compiled by the "SCHEMA DDL COMPILER".

Subschema-name

Used to identify the subschema which the run-unit is to access. It must identify a FORTRAN subschema which has been compiled by FSUBS.

FORTRAN-label

Refers to a local FORTRAN statement label which can be accessed by a local GO TO.

Integer

Used in FDML statements to provide a literal integer value. The integer syntax component obeys the normal FORTRAN rules for an integer constant.

Literal

Used in the FDML statements to refer to a literal value that is to be passed for a PRIVACY KEY statement. The literal component must refer to a valid FORTRAN integer, real or ASCII string constant.

Identifier

Used in FDML statements to refer to a FORTRAN variable name or array name.

Data-base-data-name (dbdn)

Refers to a storage location which has been declared by the schema and the subschema to hold a particular value such as an area name or data base-key.

Data-base-identifier (dbid)

Refers to a storage location which has been declared by the schema and the subschema to hold the value of an item of a record. The dbid is a valid FORTRAN identifier and is declared in the FORTRAN application program in the User Work Area COMMON block.

Area-name

Refers to a valid area name which has been defined by the schema. It is formed by providing the identifier which has been defined in the invoked subschema.

Record-name

Refers to the database record type which has been declared in the schema. It is formed by providing the identifier which has been defined in the invoked subschema.

Set-name

Refers to the database set type which has been declared in the schema. It is formed by providing the identifier which has been defined in the invoked subschema.

FDML Syntax Notation

This subsection defines the conventions used to describe syntactically correct FDML statements.

A syntax skeleton consists of a sequence of syntax component names, delimiters, and keywords, which may be grouped by special notational symbols. The skeleton represents a source specification of input to FDML in which the sequence of actual syntax components, delimiters, and keywords corresponds to that in the skeleton. The special notational symbols denote whether optional, alternative, or repeated sequences may occur within the corresponding source specification.

The following conventions apply within a syntax skeleton:

1. All underlined upper-case words are required when the format is used.

2. Upper-case words which are not underlined are optional words and need not be used.

3. Lower-case words are generic terms which must be replaced by appropriate names or values.

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix}$$ A source specification is not required in this position but it may contain either a, b, or c.

$$\begin{Bmatrix} a \\ b \\ c \end{Bmatrix}$$ A source specification in this position must include exactly one of the components a, b, or c.

$$\left\| \begin{matrix} a \\ b \\ c \end{matrix} \right\|$$ A source specification in this position may consist of a list of one or more of a, b, or c. If more than one is specified, they must be separated by commas.

...   The immediately preceding construct may be repeated an arbitrary number of times.

FDML PREPROCESSOR COMMANDS

This subsection describes in detail the FDML declarations and commands. The description of each clause contains the following headings:

Function

A brief narrative description of the function of the facility.

General Format

The arrangement of the syntax elements which make up the clause.

Syntax Rules

These serve to amplify or restrict the usage of the elements within the general format.

General Rules

1. A description of the semantic rules for using the command and a description of program capability and error conditions which relate to the statement.

Examples and Discussion

Specific examples of the use of the statement.

COMPLETE SYNTAX SKELETON

ABORT TRANSACTION identifier-1

CLEAR ERROR

CLOSE

Format 1

CLOSE ALL AREAS

Format 2

CLOSE AREA[S] area-name-1 [,area-name-2]...

DELETE $\begin{bmatrix} \text{MANDATORY} \\ \text{SELECTIVE} \\ \text{ALL} \end{bmatrix}$

END TRANSACTION identifier-1

EXIT DBMS

$$\left\{ \begin{matrix} \underline{\text{FETCH}} \\ \underline{\text{FIND}} \end{matrix} \right\} \text{rse}$$

Record Selection Expressions

Format 1

USING identifier-1

Format 2

$$\left[\left\{ \begin{matrix} \underline{\text{OWNER}} \\ \underline{\text{MEMBER}} \end{matrix} \right\} \text{IN set-name OF} \right] \underline{\text{CURRENT}} \text{ OF} \left\{ \begin{matrix} \underline{\text{RECORD}} \text{ record-name-2} \\ \underline{\text{SET}} \text{ set-name-4} \\ \underline{\text{AREA}} \text{ area-name-1} \\ \underline{\text{RUN}}\text{-UNIT} \end{matrix} \right\}$$

Format 3

$$\left\{ \begin{matrix} \underline{\text{NEXT}} \\ \underline{\text{PRIOR}} \\ \underline{\text{FIRST}} \\ \underline{\text{LAST}} \\ \text{integer-1} \\ \text{identifier-2} \end{matrix} \right\} \underline{\text{RECORD}} \text{ [record name-3] OF} \left\{ \begin{matrix} \underline{\text{SET}} \text{ set-name-5} \\ \underline{\text{AREA}} \text{ area-name-2} \end{matrix} \right\}$$

Format 4

[NEXT DUPLICATE WITHIN] RECORD record-name-4

Format 5

record-name-5 VIA [CURRENT OF] SET set-name-7 [USING dbid-3 [,dbid4]...]

Format 6

NEXT DUPLICATE WITHIN SET set-name-8 USING dbid-5 [,dbid-6]...

GET [dbid-1 , dbid-2...]

IF

Format 1

IF set-name-1 SET [NOT] EMPTY $\left\{ \begin{array}{l} \text{fortran-label-1} \\ \text{NEXT} \end{array} \right\}$ $\left[\underline{\text{ELSE}} \text{ fortran-label-2} \right]$

Format 2

IF RECORD [NOT] $\left\{ \begin{array}{l} \text{MEMBER} \\ \text{OWNER} \end{array} \right\}$ OF $\left\{ \begin{array}{l} \text{set-name-2} \\ \text{ANY SET} \end{array} \right\}$ $\left\{ \begin{array}{l} \text{fortran-label-3} \\ \text{NEXT} \end{array} \right\}$

[ELSE fortran-label-4]

INSERT INTO $\left\{ \begin{array}{l} \underline{\text{SETS}} \text{ set-name-1 } [,\text{set-name2}]... \\ \underline{\text{ALL}} \text{ SETS} \end{array} \right\}$

INVOKE DBMS

MODIFY   [dbid-1 , dbid-2...]

MOVE

Format 1

$$\text{MOVE CURRENCY STATUS FOR} \left\{ \begin{array}{l} \text{RUN-UNIT} \\ \text{RECORD record-name} \\ \text{AREA area-name} \\ \text{SET set-name} \end{array} \right\} \text{TO identifier-1}$$

Format 2

$$\text{MOVE} \left\{ \begin{array}{l} \text{RECORD-NAME} \\ \text{AREA-NAME} \end{array} \right\} \text{FOR} \left\{ \begin{array}{l} \text{RUN-UNIT} \\ \text{RECORD record-name} \\ \text{AREA area-name} \\ \text{SET set-name} \\ \text{identifier-2} \end{array} \right\} \text{TO identifier-3}$$

ON ERROR

Format 1

;ON ALL ERRORS GO TO fortran-label

Format 2

;ON ERROR integer-1 [,integer-2]... GO TO fortran-label ...

   [ON ERROR integer-3[,integer-4] ... GO TO fortran-label] ...

   [ON OTHER ERRORS GO TO fortran-label]

OPEN

Format 1

OPEN ALL AREAS $\left[\underline{\text{USAGE-MODE}} \text{ IS } \left[\begin{array}{c}\text{EXCLUSIVE} \\ \text{PROTECTED}\end{array}\right]\left\{\begin{array}{c}\text{RETRIEVAL} \\ \text{UPDATE}\end{array}\right\}\right]$

Format 2

OPEN AREA [S] area-name-1 [,area-name-2]...

$\left[\underline{\text{USAGE}} \text{ MODE IS } \left[\begin{array}{c}\text{EXCLUSIVE} \\ \text{PROTECTED}\end{array}\right]\left\{\begin{array}{c}\text{RETRIEVAL} \\ \text{UPDATE}\end{array}\right\}\right]$

PRIVACY KEY

Format 1

PRIVACY KEY [FOR $\left\|\begin{array}{c}\left[\begin{array}{c}\text{EXCLUSIVE} \\ \text{PROTECTED}\end{array}\right] \text{RETRIEVAL} \\ \left[\begin{array}{c}\text{EXCLUSIVE} \\ \text{PROTECTED}\end{array}\right] \text{UPDATE}\end{array}\right\|$]OF

$\left\{\begin{array}{l}\underline{\text{AREAS}} \text{ area-name-1 } [,\text{area-name-2}]... \\ \underline{\text{ALL}}\ \underline{\text{AREAS}}\end{array}\right\}$ $\underline{\text{IS}}$ $\left\{\begin{array}{l}\text{literal-1} \\ \text{identifier-1}\end{array}\right\}$

Format 2

PRIVACY KEY [FOR $\left\{\begin{array}{l}\text{REST} \\ \text{STORE} \\ \text{GET} \\ \text{MODIFY} \\ \text{INSERT} \\ \text{REMOVE} \\ \text{DELETE MANDATORY} \\ \text{DELETE SELECTIVE} \\ \text{DELETE ALL} \\ \text{FIND}\end{array}\right\}$] OF

$\left\{\begin{array}{l}\underline{\text{RECORDS}} \text{ record-name-1 } [,\text{record-name-2}]... \\ \underline{\text{ALL}}\ \underline{\text{RECORDS}}\end{array}\right\}$ $\underline{\text{IS}}$ $\left\{\begin{array}{l}\text{literal-2} \\ \text{identifie}\end{array}\right\}$

Format 3

$$\text{PRIVACY } \underline{\text{KEY}} \text{ [FOR } \left\{ \left\| \begin{array}{l} \underline{\text{REST}} \\ \underline{\text{STORE}} \\ \underline{\text{GET}} \\ \underline{\text{MODIFY}} \end{array} \right\| \right\} \text{]OF DATA-}\underline{\text{ITEMS}} \text{ dbid-1 [,dbid-2]...}$$

$$\underline{\text{IS}} \left\{ \begin{array}{l} \text{literal-3} \\ \text{identifier-3} \end{array} \right\}$$

Format 4

$$\text{PRIVACY } \underline{\text{KEY}} \text{ [FOR } \left\{ \left\| \begin{array}{l} \underline{\text{REST}} \\ \underline{\text{INSERT}} \\ \underline{\text{REMOVE}} \\ \underline{\text{FIND}} \end{array} \right\| \right\} \text{] OF } \left\{ \begin{array}{l} \underline{\text{SETS}} \text{ set-name-1 [,set-name-2]...} \\ \underline{\text{ALL SETS}} \end{array} \right\}$$

$$\underline{\text{IS}} \left\{ \begin{array}{l} \text{literal-4} \\ \text{identifier-4} \end{array} \right\}$$

$$\underline{\text{REMOVE}} \text{ } \underline{\text{FROM}} \left\{ \begin{array}{l} \underline{\text{SETS}} \text{ set-name-1 [,set-name-2]...} \\ \underline{\text{ALL}} \text{ SETS} \end{array} \right\}$$

$$\underline{\text{START TRANSACTION}} \text{ identifier } \left[\begin{array}{l} \underline{\text{UPDATE}} \\ \text{,}\underline{\text{RETRIEVAL}} \end{array} \right]$$

STORE record-name

SUBSCHEMA subschema-name of SCHEMA schema-name

SUPPRESS

$$
\text{[CLEAR]} \left[\text{SUPPRESS} \left\{ \left\| \begin{array}{l} \text{ALL} \\ \text{RECORD} \\ \text{AREA} \\ \text{SET} \\ \text{set-name-1 [set-name-2]...} \end{array} \right\| \right\} \right]
$$

DML STATEMENTS

```
********************
* ABORT TRANSACTION *
********************
```

Function

Causes all update actions since the last START OF TRANSACTION command to be negated (i.e., the database is rolled back), all locks to be released, and all currency indicators to be reset as they were at the last START TRANSACTION command.

General Format

ABORT TRANSACTION identifier-1

Syntax Rules

1. Identifier-1 must contain the value assigned by the START OF TRANSACTION command.

General Rules

1. If before-imaging is turned off then the ABORT TRANSACTION has no effect.

2. If identifier-1 is invalid or if no transaction is currently active, Error Status Condition 2535 results with concurrent Error Status found in CONTYP.

Examples and Discussion

See START TRANSACTION.

```
***************
*  CLEAR ERROR  *
***************
```

Function

CLEAR ERROR CLEARS DMLCP error conditions and its associated registers.

General Format

    CLEAR ERROR

Syntax Rules

None

General Rules

1. This command clears all DBMS registers, including ERSTAT, CONTYP, ERAREA, ERDEC, ERSET, ERITEM, ERCASE.

2. If the error is a non-fatal error, it clears on internal flag allowing the DMLCP to continue processing.

3. It must be called after every DMLCP error condition.

Examples and Discussion

See ON ERROR clause.

```
*********
* CLOSE *
*********
```

Function

To relinquish control over the specified areas and make them available to other run-units.

General Format

Format 1

    CLOSE ALL AREAS

Format 2

    CLOSE AREA[S] area-name-1 [,area-name-2]...

Syntax Rules

1. All area names specified must be included in the invoked subschema.

General Rules

1. Note that CLOSE (unlike OPEN) need not deal with all opened areas.

2. After execution of a CLOSE statement for a given area, any attempt to access that AREA will result in Error Status Condition nn01 (where nn indicates the particular DML statement attempted). A subsequent CLOSE statement does not return an error status.

3. In format 1, the CLOSE statement applies to all areas which are in an open status for the run-unit. These areas are then subject to General Rule 2 above.

4. The run-unit's currency indicators may identify those record occurrences located in areas which are no longer in an open status for that run-unit. These currency indicators become null after the CLOSE statement is executed.

5. If all area-names specified are not included in the invoked subschema, Error Status Condition 0146 results; otherwise ERSTAT is zero, indicating successful execution of the CLOSE statement.

6. A CLOSE statement cannot be executed during an open transaction. Otherwise, Error Status Condition 935 will result, with CONTYP set to 15.

Examples and Discussion for the CLOSE Statement

Example 1

    #CLOSE AREAS MY-AREA, YOUR-AREA.

This statement closes two areas: 1) MY-AREA and 2) YOUR-AREA. All
records in these areas are no longer available for processing and all
of the currency statuses that refer to records in those areas are
cleared.

```
**********
* DELETE *
**********
```

Function

The DELETE statement performs the following functions:

1. Makes the object record occurrence unavailable for further processing by the imperative-statements of the DML.

2. Removes the object record from all set occurrences in which it is a member.

3. Deletes all record occurrences which are mandatory members of set occurrences owned by the object record.

4. Removes or deletes optionally all record occurrences which are optional members of set occurrences owned by the object record.

5. Optionally prevents deletion of the object record if the database contains any non-empty set occurrences of which the object record is the owner.

General Format

$$\text{DELETE} \quad \begin{bmatrix} \underline{\text{MANDATORY}} \\ \underline{\text{SELECTIVE}} \\ \underline{\text{ALL}} \end{bmatrix}$$

Syntax Rules

None.

General Rules

1. The immediate object record occurrence of the DELETE statement is the current record of the run-unit.

2. The object record is removed from all set occurrences in which it is a member. It is then deleted; that is, made unavailable for further processing by any DML statement.

   Note that there is a distinction made between a removed record occurrence and a deleted record occurrence. A DELETE statement

is inclusive of the function of a REMOVE statement, and is described in the previous paragraph of this general rule. A REMOVE statement cancels the existing membership of a record occurrence in specific set occurrences. The removed record is then not accessible through those set occurrences but continues to be accessible through any other sets in which it participates as a member. It may also be accessible by virtue of its having been defined with a LOCATION MODE IS CALC clause. It is always accessible by means of a complete scan of the area in which it participates, or through its database-key (if that is known).

3. The unqualified form of the DELETE statement deletes the object record only if it has no member records. If the object record is the owner of a non-empty set occurrence, the DELETE statement is not successfully executed and Error Status Condition 0230 results.

4. The DELETE MANDATORY form of the statement deletes the object record and all of its mandatory members. It removes but does not delete its optional members.

    If any deleted mandatory member is the owner of a set occurrence, then the DELETE statement is executed on such records as if it were the object record of a DELETE MANDATORY statement. Thus, all mandatory members of such sets are also deleted, which in turn causes this process to continue down the hierarchy.

5. The DELETE SELECTIVE form of the statement has the same results as the DELETE MANDATORY statement with the exception that:

    o   Optional members are deleted if they do not currently participate as members in other set occurrences.

    o   All deleted records which are themselves the owners of any set occurrences are treated as if they were the object of a DELETE SELECTIVE statement.

6. The DELETE ALL form of the statement deletes the object record together with all of its member records, regardless of whether they are mandatory or optional. As with the DELETE MANDATORY form of the statement, the delete process continues down the hierarchy with the difference that all deleted records, which are themselves the owners of any set occurrences, are treated as if they were the object record of a DELETE ALL statement.

7. The current record of the run-unit becomes null. No other currency status information is altered. Thus, the object record and all other records deleted or removed remain as current of area-name, record-name, and of all set-names in which they were current prior to the execution of the DELETE statement.

8. If the run-unit has not satisfied the privacy locks for the object record itself, or, for any other record occurrence which

would be deleted, removed, or modified as a result of the execution of the DELETE statement, Error Status Condition 0204F will result and the run-unit will be aborted by DMLCP.

9. If the areas in which the records are being deleted are opened for concurrent update and this command causes a concurrent update conflict, or no update transaction is currently active, then error 235 will result with concurrent error status found in CONTYP.

10. In addition to the conditions described in General Rule 8, any of the following conditions will result in an error if they occur during the execution of a DELETE statement:

    o   If the current record of run-unit is not known, Error Status Condition 0213 results.

    o   If the unqualified form of the DELETE statement is attempted against the owner record of a non-empty set occurrence, Error Status Condition 0230 results.

    o   If any of the record occurrences which would be deleted, removed, or modified as a result of the execution of the DELETE statement is in an area which is not open, Error Status Condition 0201 results.

    o   If the object record, or any record that would be deleted or removed as a result of executing the DELETE statement, is located within an area that is open for RETRIEVAL, Error Status Condition 0209 results.

    o   If any record occurrence needed by the DMLCP for informational purposes (such as following a search path) is not available because it is off-line or under exclusive control of another run-unit, Error Status Condition 0218 results.

    o   The subschema invoked must name:

        a. All of the records which would be deleted, removed or modified as a result of executing the DELETE statement.

        b. All of the sets in which any record to be deleted is an owner or a mandatory member,

        c. All of the sets from which any record is to be removed.

        d. All of the owner record types of the sets from which records are being removed. Otherwise, Error Status Condition 0208 results.

11. When an error occurs, the database and User Work Area remain in the state existing prior to the attempted execution of the DELETE

statement and the appropriate Error Status Condition code is made
available in special register ERSTAT. Otherwise, ERSTAT is set
to zero, indicating successful execution of the DELETE statement.

Error Status Codes For DELETE Statement

Condition	Content of Error Status
Area not open	0201
Referenced record-name or set-name not in invoked sub-schema	0208
Incorrect usage mode for area	0209
No current record of run-unit	0213
Implicitly referenced area not available	0218
Concurrent update conflict	0235
Unqualified DELETE command attempted on non-empty set	0230
Fatal - privacy breach attempted	-0204(0204F)

Examples and Discussion for the DELETE Statement

Example 1

    #DELETE ALL.

    This statement will delete the record that is the current record of
    run-unit and all of its members and remove it from all sets in
    which it is a member. It will then repeat this process until all
    of the records which participate in the hierarchy have been
    deleted.

Example 2

    #DELETE.

    This statement will delete the current record at run-unit only if
    it owns no sets or if the sets which it does own currently have no
    members. It also removes the record from all sets in which it
    participates as a member.

```
*******************
*  END TRANSACTION *
*******************
```

Function

END TRANSACTION defines the end of a recoverable program sequence and causes a record of the successful termination to be written to the DBMS system LOG file.

The DBMS guarantees that all updates are secure to disk.

The end of an update transaction allows modified data to be accessed by other transactions by releasing all locks.

General Format

    END TRANSACTION identifier-1

Syntax Rules

1.   Identifier-1 is a FORTRAN identifier which contains an INTEGER*2 value assigned to the transaction.

General Rules

1.   If before-imaging is turned off, then the END TRANSACTION has no effect.

2.   If identifier-1 does not match the identifier on the most recent START TRANSACTION or if no transaction is currently active, Error Status Condition 2435 results with concurrent error status found in CONTYP.

Examples and Discussion

(See START TRANSACTION).

```
*************
* EXIT DBMS *
*************
```

Function

To terminate communication with the DMLCP. To close all opened files
and clear all status registers.

General Format

EXIT DBMS

Syntax Rules

None

General Rules

1. An EXIT DBMS must be executed before termination of a run-unit.

2. Once an EXIT DBMS has been executed, the only way communication
    to the DMLCP can be reinitialized is through an INVOKE.

3. An EXIT DBMS cannot occur inside an opened DBMS transaction.
    Otherwise, an error 1035 results with concurrent error status of
    15 in CONTYP.

Examples and Discussion

NONE

```
*********
* FETCH *
*********
```

Function

The FETCH statement is a combined FIND and GET statement which both establishes currency for the object record and makes available the contents of that record.

General Format

FETCH rse

Syntax Rules

See RECORD SELECTION EXPRESSION (rse).

General Rules

1. Execution of a FETCH statement causes the affected record to become available to the program; that record becomes the current record of the run-unit. A FETCH command is logically identical to execution of a FIND followed by a GET of all data items in that record.

2. Refer to the discussion of the FIND statement.

3. Refer to the discussion of the GET statement.

```
********
* FIND *
********
```

Function

The FIND statement specifies a record occurrence as:

1. The current record of the run-unit.

2. The current record of the area in which it is stored.

3. The current record of its record-name.

4. The current record of set for all set-names in which it currently participates as owner or member.

A prior SUPPRESS may prevent the establishment of the object record occurrence as:

1. The current record of the area in which it is stored.

2. The current record of its record-name.

3. The current record of set for all, or specified sets in which it currently participates as an owner or member.

General Format

    FIND rse

Syntax Rules

See RECORD SELECTION EXPRESSION (rse)

NOTE:

Refer to rse for a discussion of the record-selection-expression.

General Rules

1. Execution of a FIND statement causes the record referenced by the record-selection-expression to become the current record of the run-unit.

2. Execution of a FIND statement does not make the selected record available to the program - it merely identifies the record for

use in certain subsequent statements. These include: GET, MODIFY, INSERT, REMOVE and DELETE.

3. If the suppression of updating currency indicators has not been set, the object record also becomes the current record of its area-name, the current record of its record-name, and the current record of all set-names in which it is defined as an owner or in which it currently participates as a member.

4. The effect of a SUPPRESS command or a series of SUPPRESS commands is to specify the area, record, set, and set-name currency indicators which are to retain their existing status.

5. If any of the following conditions are encountered, the FIND statement is not successfully executed, the database remains in the state existing prior to the attempted execution, and the appropriate Error Status Condition code is made available in special register ERSTAT. Otherwise, ERSTAT is set to zero, indicating successful execution of the FIND statement.

    o   If the sought record is in an area which has not been opened, Error Status Condition 0301 results.

    o   If any record occurrence along the search path of the FIND statement is in areas which are off-line or under the exclusive control of a concurrent run-unit, Error Status Condition 0318 results.

    o   If any current record of the type specified is not known, Error Status Condition 0306 results.

    o   If a database-key is supplied or developed which is incompatible with the areas specified, Error Status Condition 0302 results.

    o   If an end-of-set, end-of-area, or end-of-Dups condition is encountered, Error Status Condition 0307 results.

    o   If no record in the area satisfies the FIND specified, Error Status Condition 0326 results.

    o   If a specific record type is sought in a set occurrence or is used in a FIND to define the current set occurrence of a set, and the record type is not defined as a member of the set in the schema, Error Status Condition 0340 results.

    o   If AREA-ID is initialized with an area-name not included in the WITHIN clause in the schema DDL, Error Status Condition 0323 results.

    o   All data items, areas, records, and sets specified in a FIND statement must be defined in the invoked subschema; otherwise, Error Status Condition 0346 results.

Error Status Codes For the FIND Statement

Condition	Content of ERROR-STATUS
Area not open	0301
Database-key inconsistent with area-name	0302
Data items invalid or inconsistent	0304
Current record of set, area, record-name not known	0306
End of set or area or DUPs	0307
Implicitly referenced area not available	0318
Record not currently a member of named set	0322
Illegal area-name	0323
No record satisfies the FIND specified	0326
Concurrent access error	0335
Attempted to find owner of a singular set	0333
Record type not a member of named set	0340
Record type not owner of named set	0341
Record type not included in named area	0342
Arguments of Location Mode Clause not included in subschema	0343
Arguments of Set Occurrence Selection Clause not included in subschema	0344
Location mode of record not specified as CALC	0345
Specified data item, record, set, or area not in subschema	0346
Fatal - privacy breach attempted	-0304(0304F)

.

.

Examples and Discussion of the FIND Statement

See the Examples and Discussion under Record Selection Expression.

```
*******
* GET *
*******
```

Function

Transfers the contents of the specified data items of the object record occurrence into the User Work Area for the invoked subschema.

General Format

    GET [dbid-1 , dbid-2...]

Syntax Rules

1. Dbid-1, dbid-2..., must be items defined in the subschema as being in the record type of the current record of the run-unit.

General Rules

1. The object of the GET statement is the current record of the run-unit.

2. The record-name of the object record serves as an implicit major qualifier for the data-base-identifiers. If they are not defined as part of the record type, Error Status Condition 0504 results.

3. In cases where the format of a data item, as defined in the subschema invoked by the run-unit, differs from the definition of that data item in the schema:

   o The standard DMLCP procedure will be in accordance with the rules specified in the TYPE clause of the Schema DDL and the TYPE specification of the Subschema DDL for FORTRAN.

4. Missing data items in the object record occurrence in the database will result in null-values being placed, for those data items, in the User Work Area. The value of a null item is as follows:

   o For items typed numeric, the item is set to zero.

   o For items typed string, the item is set to blanks.

   o For items typed logical, the item is set FALSE.

5. A GET statement must be executed before any reference can be made to the data of the object record in the User Work Area.

6. If only "GET" is written, then all data items defined for the object record in the subschema invoked by the run-unit are moved to the User Work Area. If dbid's are specified, only the specified data items are moved to the User Work Area.

7. If the current record of the run-unit is not known, Error Status Condition 0513 results.

8. If the run-unit has not satisfied the privacy locks on all data items and records needed to execute the GET, Error Status Condition 0504F will result and the run-unit will be aborted by DMLCP.

9. If a dbid which is implied or specified in the GET statement is dimensioned in the invoked subschema, DMLCP will place a value in each element of the dimensioned item. If the dimension in the User Work Area is greater than the number of multiple values of the item in the object record occurrence, the excess elements of the item in the User Work Area will be set to the appropriate null-value (see general rule 4).

10. If the value of any data item in the database is such that it cannot be converted to the specified subschema format for that data item, Error Status Condition 0519 results. (Conversion rules are included under the description of TYPE clause in the DDL for the SCHEMA and TYPE specification in the subschema DDL for FORTRAN.)

11. If any of the above Error Status Conditions is encountered, then the GET statement is not successfully executed. The record in the User Work Area is in an undefined state and the appropriate Error Status Condition code is made available in special register ERSTAT. Otherwise, ERSTAT is set to zero, indicating successful completion of the GET statement.

Error Status Codes For The GET Statement

Condition	Content of ERROR-STATUS
Data item invalid or inconsistent	0504
No current record of run-unit	0513
Conversion of value of data item not possible	0519
Concurrent access error	0535
Specified Database identifier not included in subschema	0546
Fatal - privacy breach attempted	-0504(0504F)

Examples and Discussion of the GET Statement

    #GET LNAME, FNAME, MNAME, DOB

This command moves the contents of LNAME, FNAME, MNAME, DOB from the
record to the User Work Area. It assumes that the current record of
run-unit has these items defined within it and that the subschema has
declared them. If necessary, conversion is performed on these items to
change them from the schema type to the subschema type. Only the named
items will be obtained from the record. The rest of the variables in
the User Work Area are left unchanged.

    #GET

This command gets all the data items from the schema for the record
type which is current of run-unit and places them in the User Work
Area.

```
******
* IF *
******
```

Function

The IF statement causes a condition to be evaluated. The subsequent action of the run-unit depends on whether the value of the condition is true or false.

General Format

Format 1

$$\text{IF set-name-1 } \underline{\text{SET}} \ [\underline{\text{NOT}}] \ \text{EMPTY} \left\{ \begin{matrix} \text{fortran-label-1} \\ \underline{\text{NEXT}} \end{matrix} \right\} \left[\underline{\text{ELSE}} \ \text{fortran-label-2} \right]$$

Format 2

$$\text{IF } \underline{\text{RECORD}} \ [\underline{\text{NOT}}] \left\{ \begin{matrix} \underline{\text{MEMBER}} \\ \underline{\text{OWNER}} \end{matrix} \right\} \text{OF} \left\{ \begin{matrix} \text{set-name-2} \\ \underline{\text{ANY}} \ \text{SET} \end{matrix} \right\} \left\{ \begin{matrix} \text{fortran-label-3} \\ \underline{\text{NEXT}} \end{matrix} \right\}$$

$$[\underline{\text{ELSE}} \ \text{fortran-label-4}]$$

Syntax Rules

1. Set-name-1, Set-name-2, must be defined in the invoked subschema.

General Rules

1. Set-name-1, set-name-2 must be defined in the invoked subschema; otherwise, Error Status Condition 1846 results.

2. Format 1 of the IF statement is designed to determine whether the object set occurrence has any members. The object set occurrence is determined by the current record of set-name-1. If the NOT phrase is omitted and the set occurrence does not have any member records, the condition is evaluated as true. If the NOT phrase is omitted and the set occurrence contains member records, the condition is evaluated as false. If the NOT phrase is stated, the condition is reversed.

3. Format 2 of the IF statement is designed to determine whether the

current record of the run-unit currently participates as an owner or member, depending on the option specified, in set-name-2 or in any set. If the NOT phrase is omitted and the record is an owner or member as specified, the condition is evaluated as true. If the NOT phrase is omitted and the record is not an owner or member as specified, the condition is evaluated as false. If the NOT phrase is stated, the condition is reversed. If neither the OWNER nor MEMBER phrase is specified, the test is of the association as an owner or member of the object record with the specified set.

4. If the current record of the run-unit is not known, Error Status Condition 1813 results.

5. If the current record of the set-name specified is not known or the currency indicator is null, then the object set occurrence cannot be determined and Error Status Condition 1806 results.

6. If any of the above Error Status Conditions is encountered, then the IF statement is not successfully executed. The condition stated in the IF statement will be left unevaluated and the appropriate Error Status Condition code is made available in special register ERSTAT.

Error Status Codes for the IF Statement

Condition	Content of ERSTAT
Current of set not known	1806
No current record of run-unit	1813
Specified set not included in subschema	1846

Examples and Discussion of the IF Statement

Example 1

```
        #IF DEPT-SET SET EMPTY NEXT ELSE 99
            .
            .
            .
            .
            .
   99    CONTINUE
```

The IF command allows the application programmer to test for a number
of conditions within the DBMS. The test in example one checks to see
if there are any members of the "DEPT-SET". If the set is empty,
control passes to the next statement; otherwise, control passes to
statement 99.

Example 2

```
        #IF RECORD NOT MEMBER OF DEPT-SET 100.
```

In this example, the current record of run-unit is tested to see if it
is a member of the DEPT-SET. If it is, control is passed to the next
statement; otherwise, control is passed to statement 100.

```
**********
* INSERT *
**********
```

Function

Inserts and makes the object record a member of occurrences of the specified set-names, provided that it is defined as an optional automatic, optional manual, or mandatory manual member of those sets.

General Format

$$\text{INSERT INTO} \left\{ \begin{array}{l} \underline{\text{SETS}} \text{ set-name-1 [,set-name2]...} \\ \underline{\text{ALL}} \text{ SETS} \end{array} \right\}$$

Syntax Rules

1. The sets set-name-1, set-name-2, must be defined in the invoked subschema.

2. The record that is the current record at run-unit, must be defined as an optional or mandatory manual member of the specified sets. If the ALL option is used, the record must be a member of at least one of the sets in the invoked subschema.

General Rules

1. The object record occurrence of the INSERT statement is the current record of the run-unit.

2. If set-names are specified, then the object record must have been defined in the schema as an optional automatic, optional manual, or mandatory manual member of each set named. It will be inserted into the object set occurrence of each set-name specified in accordance with the set-ordering criteria defined in the schema. For each set named, the object set occurrence is determined by the current record of the named set.

o   If the current record of any set-name specified is not known
     or its currency indicator is null, Error Status Condition
     0706 results.

3. The ALL SETS option inserts the object record into the
     appropriate occurrence of each set included in the invoked
     subschema, where it is defined as an optional automatic, optional
     manual, or mandatory manual member and in which it is not
     currently a member. The specific occurrence of each set will be
     determined by the current record of set-name for each of the
     set-names involved. The object-record will be inserted into each
     set occurrence in accordance with the set-ordering criteria
     specified in the schema.

     o   If the current record of any set-name implicitly specified by
          ALL SETS is not known or its currency indicator is null,
          Error Status Condition 0706 results.

4. For each set-name into which the object record is inserted, it
     becomes the current record of set-name if such a currency update
     is not in a 'SUPPRESS' state (see the SUPPRESS command).

5. If the area in which the inserted record is stored is opened for
     concurrent update and there is a concurrent update conflict,
     Error Status Condition 735 results with CONTYP set to the
     concurrent update conflict type.

6. If the run-unit has not satisfied the privacy locks for all
     records, sets, areas, and data items needed to execute the INSERT
     statement, Error Status Condition 0704F will result and the
     run-unit will be aborted by DMLCP.

7. In addition to the conditions described, any of the following
     conditions will result in an error if they occur during the
     execution of an INSERT statement:

     o   If all set-names specified are not included in the subschema,
          Error Status Condition 0746 results.

     o   If the current record of the run-unit is not known, Error
          Status Condition 0713 results.

     o   If the object record is not defined as an optional automatic,
          optional manual, or mandatory manual member of all of the
          specified set-names, Error Status Condition 0714 results.

     o   If the object record, when inserted, would violate a
          DUPLICATES NOT ALLOWED clause for any record or set involved,
          Error Status Condition 0705 results.

     o   If the current record of any set-name specified is not known
          or its currency indicator is null, Error Status Condition
          0706 results.

o   If the object record is already a member of any occurrence of
    a set explicitly named in the INSERT statement, or if it is
    already a member of an occurrence of each set implicitly
    specified by the ALL SETS option, Error Status Condition 0716
    results. This is true whether the object record is a member
    in the object set occurrence or in any other occurrence of
    the same set.

o   If the object record or any record occurrence affected by the
    INSERT statement is located in an area which is open for
    RETRIEVAL, Error Status Condition 0709 results.

o   If any record occurrence needed to execute the operation is
    located in areas that are not available, Error Status
    Condition 0718 results. An area is not available if it is
    off-line or under the exclusive control of a concurrent
    run-unit.

8. When an error occurs, the database remains in the state existing
    prior to the attempted execution of the INSERT statement and the
    appropriate Error Status Condition code is made available in
    special register ERSTAT.

Error Status Codes for the INSERT Statement

Condition	Content of ERROR-STATUS
Violation of DUPLICATES NOT ALLOWED clause	0705
Current record of set-name not known	0706
Incorrect usage mode for area	0709
No current record of run-unit	0713
Object record not defined as an optional member or mandatory manual member of a named set	0714
Record already a member of named set	0716
Implicitly referenced area not available	0718
Concurrent update conflict	0735
Specified set-name not in subschema	0746
Fatal – privacy breach attempted	-0704(070CF)

Examples and Discussion for the INSERT Statement

    ‡INSERT INTO ALL SETS.

The current record of run-unit is inserted into all sets of which it is
not already a member. In addition, if the sets have search keys
defined, the search key is inserted into its list. The insertion into
the set is based on the ordering criteria given in the schema.

```
**********
* INVOKE *
**********
```

Function

INVOKE is used to indicate that the FORTRAN program is ready to use the DMLCP services and to establish the program as a DBMS user.

It is also used to invoke the use of the subschema specified in the subschema declaration.

General Format

<div align="center">

INVOKE DBMS

</div>

Syntax Rules

None.

General Rules

1. The INVOKE statement must be executed before any other DML statement. Before it is executed, all other DML statements executed by the FORTRAN program will be ignored by DBMS.

2. The INVOKE statement causes a connect sequence to establish the FORTRAN program as a run-unit under Prime DBMS.

3. If the schema or the subschema which the run-unit wishes to use is not available to the DBMS, a fatal error occurs and the run-unit will be aborted.

Error Status Codes for the INVOKE Statement

Condition	Content of ERSTAT
Invoke has already been executed	1403
Fatal - invoked subschema has been deleted from DBMS	-1401(1401F)
Fatal - invoked schema has been deleted from DBMS	-1402(1402F)

Examples and Discussion

None.

```
**********
* MODIFY *
**********
```

Function

The MODIFY statement replaces the values of all, or of specific data items of the object record occurrence in the database, with values from the User Work Area.

General Format

MODIFY   [dbid-1 , dbid-2...]

Syntax Rules

1. dbid-1, dbid-2..., must be defined as items of the record that is the current record at run-unit.

General Rules

1. The object record occurrence of the MODIFY is the current record of the run-unit. If the current record of the run-unit is not known, the MODIFY statement is not executed and Error Status Condition 0813 results.

2. The record-name of the object record serves as an implicit major qualifier for the data-base-identifiers. They must be defined as part of the record type, otherwise, Error Status Condition 0804 results. 0804 results.

3. If data-base-identifiers are not specified, all data items in the object record which are specified in the subschema named in the INVOKE clause of the program are modified with (i.e., replaced by) values from the User Work Area. Any data item in the object record occurrence in the database which is not specified in the invoked subschema remains unchanged.

4. If data-base-identifiers are specified, only the data items specified are modified with values from the User Work Area. All other data items in the object record occurrence in the database remain unchanged.

5. In cases where the definition of a data item in the subschema invoked by the run-unit differs from the definition of that data item in the schema, conversions performed by DMLCP will be in accordance with the rules specified in the TYPE clause of the Schema DDL and the TYPE specification of the subschema DDL for FORTRAN.

6. If any of the modified data items is defined as a sort-control item in the object record for any set occurrence in which the object is currently a member, then its modification causes the intra-set occurrence position of the object record to be examined. If necessary, the object record is removed and re-inserted in the set occurrence to maintain the set order

specified in the schema. The current occurrence of the set-name involved remains as the current occurrence.

If the current of run-unit is the current of the set-name involved, it remains as the current of set-name. If it is not, it becomes the current of that set-name if such a currency update is not in a 'SUPPRESS' state (see the SUPPRESS command). The database-key of the object record remains unchanged.

7. If any of the modified data items are defined with a SEARCH KEY clause in any set in which the object record is currently a member, then the execution of the MODIFY statement causes DMLCP to adjust the indices it maintains for those data items. The database-key of the object record remains unchanged.

8. If any of the modified data items is defined as a CALC key in the LOCATION MODE clause for the object record, then the execution of the MODIFY statement causes the DMLCP to make the necessary adjustments which will enable the record to be found on the basis of the new values for the CALC keys. The data item (if any) specified as AREA-ID must also be initialized with an area-name specified or implied for the record type; otherwise, Error Status Condition 0823 results.

9. If the insertion of the object record under any of the conditions described in general rules 6, 7, or 8 would violate a DUPLICATES NOT ALLOWED clause (defined for any of the sets or records involved), then the MODIFY statement is not executed and Error Status Condition 0805 results.

10. All data items involved must be initialized in the User Work Area with the required values prior to execution of the MODIFY statement.

11. If the run-unit has not satisfied the privacy locks for all records, sets, areas, and data items needed to execute the MODIFY, Error Status Condition 0804F will result and the run-unit will be aborted by DMLCP.

12. If the object record or any record occurrence affected by the execution of the MODIFY statement is located in an area which is open for RETRIEVAL, Error Status Condition 0809 results.

13. If the area in which the modified record is opened for concurrent update and there is a concurrent update conflict, Error Status Condition 0835 results and CONTYP is set to the concurrent update conflict type.

14. If any record occurrence needed to execute the operation is located in areas that are not available, Error Status Condition 0818 results. An area is not available if it is off-line or under the exclusive control of a concurrent run-unit.

15. If the value of a data item in the User Work Area is such that it cannot be converted to the format specified in the schema for that data item, Error Status Condition 0819 results.

16. Under all Error Status Conditions described in the above general rules, the

database remains in the state existing prior to the attempted execution of
the MODIFY statement, and the appropriate Error Status Condition code is
made available in special register ERSTAT. Otherwise, ERSTAT is set to
zero, indicating successful execution of the MODIFY statement.

Error Status Codes for the MODIFY Statement

Condition	Content of ERROR-STATUS
Database-key inconsistent with area-name	0802
Data items invalid or inconsistent	0804
Violation of DUPLICATES NOT ALLOWED clause	0805
Incorrect usage mode for area	0809
No current record of run-unit	0813
Implicitly referenced area not available	0818
Conversion of value of data item not possible	0819
Illegal area-name	0823
Concurrent update conflict	0835
Arguments of Location Mode clause not included in subschema	0843
Arguments of Set Occurrence Selection clause not included in subschema	0844
Specified set not included in subschema	0846
Fatal - privacy breach attempted	-0804(0804)

Examples and Discussion of the MODIFY Statement

Example 1

    #MODIFY LNAME, FNAME

The MODIFY command allows the application programmer to change the
contents of a given record. The MODIFY command always operates on
the current record of run-unit. In the example above, the fields
LNAME, FNAME in the record will be replaced by the contents of the
fields in the User Work Area. The items named must be defined in
the subschema and be contained in the record that is current record
of run-unit.

In addition to changing the contents of the fields, if the items
make up any part of a sort, the list is reordered to reflect the
new ordering with the new value. If the item is a CALC key, the
record is recalced to reflect the value of the new key.

Example 2

    #MODIFY

This form of the MODIFY command replaces the entire contents of the
record, which is current of run-unit, with the contents of the User
Work Area. It does all conversions called for between schema types
and subschema types. It also re-orders any set list which contains
items that are sort or search keys, and if the record has a
location mode CALC, the record is recalced to reflect the new value
of the CALC keys.

```
********
* MOVE *
********
```

Function

The MOVE statement saves the contents of the specified currency status indicators and provides a means for deriving the area-name or record-name which corresponds to a database-key.

General Format

Format 1

$$\text{MOVE CURRENCY STATUS FOR} \begin{Bmatrix} \text{RUN-UNIT} \\ \text{RECORD record-name} \\ \text{AREA area-name} \\ \text{SET set-name} \end{Bmatrix} \text{TO identifier-1}$$

Format 2

$$\text{MOVE} \begin{Bmatrix} \text{RECORD-NAME} \\ \text{AREA-NAME} \end{Bmatrix} \text{FOR} \begin{Bmatrix} \text{RUN-UNIT} \\ \text{RECORD record-name} \\ \text{AREA area-name} \\ \text{SET set-name} \\ \text{identifier-2} \end{Bmatrix} \text{TO identifier-3}$$

Syntax Rules

1. The specified record-name, area-name, or set-name, must be included in the invoked subschema.

2. Identifier-1, identifier-2, must refer to data items that are used as Database-Keys. Identifier-3 must be an integer array which is able to contain a string 30 characters long.

General Rules

1. The specified record-name, area-name, or set-name must be included in the invoked sub-schema; otherwise, Error Status Condition 1946 results.

2. Identifier-1 and identifier-2 are assumed to be three-word integer arrays (i.e., FORTRAN variables meant to contain database-keys). For more detail about the representation of a

database-key, refer to the TYPE clause in the Schema DDL and the TYPE specification in the Subschema DDL for FORTRAN (See Section 2).

3. Identifier-3 is assumed to be a FORTRAN variable which can contain a 30 character string, two characters packed per word (i.e., a 15-word INTEGER*2 array).

4. In Format 1, if the RUN-UNIT phrase is specified, the database-key for the current record of run-unit is placed in identifier-1. The current record of the run-unit is not altered. If a record-name, area-name, or set-name is specified, the database-key for the current record of record-name, area-name, or set-name is placed in identifier-1. The current record of record-name, area-name, or set-name is not altered. If the current record is not known or the currency indicator is null, Error Status Condition 1913 results.

5. Use of Format 2 causes DMLCP to return, in identifier-3, the name of the area or record which corresponds to the database-key in identifier-2 or to the currency indicator specified. If a currency indicator is specified and the current record is not known or the currency indicator is null, Error Status Condition 1913 results.

6. When an error occurs, the database, and all of the run-unit's FORTRAN variables remain in the state existing prior to the attempted execution of the MOVE statement.

Error Status Codes for the MOVE Statement

Condition	Content of ERSTAT
No current record of run-unit	1913
Specified record, area, or set not included in subschema	1946

Examples and Discussion of the MOVE Statement

Example 1

    #MOVE CURRENCY STATUS FOR RECORD EMPLOYEE TO ID1

This statement takes the database-key which is the currency status for
the record EMPLOYEE and moves it to the FORTRAN identifier ID1. The
run-unit can later refer to the record by a

    #FIND ID1.

To establish the record as current of run-unit.

Example 2

    #MOVE AREA-NAME FOR ID1 TO ANAME

This statement gets the 30-character Area-name of the DBK defined in
FD1 and moves it to the 15-word FORTRAN array ANAME.

```
*******************
* ON ERROR CLAUSE *
*******************
```

Function

The ON ERROR clause specifies where program control is to be transferred if the DBMS encounters an error during the execution of a DML command.

General Format

Format 1

    ;ON ALL ERRORS GO TO fortran-label

Format 2

    ;ON ERROR integer-1 [,integer-2]... GO TO fortran-label ...

        [ON ERROR integer-3[,integer-4] ... GO TO fortran-label] ...

        [ON OTHER ERRORS GO TO fortran-label]

Syntax Rules

1. integer-1, integer-2, ... are INTEGER*2 values of the form:

                            MMEE

    where   MM is the major code of the command

                 EE is the error code

                 The major code is an optional specification

2. No check is ever made to verify that the values specified by integer-1, integer-2... can actually occur. If an invalid code is used, no branch will ever be taken at run time.

3. Label must be a valid FORTRAN Statement label known to this program.

General Rules

1. The ON ERROR clause may be appended to any DML command.

2. If an error is detected, the DBMS returns to the statement specified by the appropriate ON ERROR clause (if any).

3. The ON ERROR clause refers only to the command of which it is a part.

4. If an error is detected and there is no error clause, control is returned to the run-unit at the statement following the DML command. The run-unit may test the register ERSTAT to see if an error occurred.   If ERSTAT is 0, then no error has occurred; otherwise, ERSTAT will be of the form MMEE

    where   MM is the major code

               EE is the error code

5. If an error has occurred, no further DML statements may be processed until a CLEAR ERROR command has been executed.

6. If another DML statement is executed before a CLEAR ERROR command is executed, or if a fatal error has occurred, the run-unit is immediately aborted and the special registers are dumped to the terminal.

7. Whenever the execution of a DML statement results in an Error Status Condition, the following information is available to the run-unit:

    o   The Error Status Condition code is available in the special FORTRAN integer variable ERSTAT.

    o   The name of the area in which the error occurred is available in the 15 word integer array ERAREA.

    o   The name of the set, if appropriate, in which the error occurred is available in the 15 word integer array ERSET.

    o   The name of the record, if appropriate, for which the error occurred is available in the 15 word integer array ERREC.

    o   If ERSTAT contains an error code of 35 which indicates a concurrent update conflict, then the register CONTYP contains a code indicating the type of concurrent update conflict.

    o   The name of the item, if appropriate, for which the error occurred is available in the 3-word integer array ERITEM.

    o   ERCASE contains the conditional value of an error return for a computed GO TO statement.

    o   The number of the current transaction can always be found in TRNO.

    o   The type of the current transaction can be found in TRTYPE where 1=update transaction and 0=retrieval transaction.

8. A summary of the major codes, error codes and concurrent update conflict codes may be found in Section 4.

Examples and Discussion for the ON ERROR Clause

Example 1

```
        # FIND NEXT RECORD EMPLOYEE OF SET DEPT-SET;
              ON ERROR 307 GO TO 100

   50    CONTINUE
              .
              .
              .
  100    WRITE (1,1001)

  1001   FORMAT('END OF SET ENCOUNTERED.')
```

This example shows a find statement with an error clause. If an end of set is encountered, control will pass to statement 100; otherwise control will pass to the next statement. If an error other than 307 occurred, no other FDML statement may be executed until a clear error command has been executed.

Example 2

```
        # FIND RECORD DEPT.

        IF (ERSTAT.NE.0) GO TO 100
              .
              .
              .
  100

  100    WRITE (1,1001) ERSTAT

  1001   FORMAT ('DBMS ERROR', I5)

        # CLEAR ERROR.
```

Example 2 demonstrates an alternate form of DBMS error detection. If the FIND in the example succeeds and there are no errors, ERSTAT will be 0 and control will proceed to the next statement. Otherwise, ERSTAT will be nonzero and control will be passed to statement 100 where the error is written to the user terminal and the error status is reset by the CLEAR ERROR command.

```
********
* OPEN *
********
```

Function

The OPEN statement specifies an area's usage-mode and prevents the run-unit's access to the area until such usage can be permitted.

General Format

Format 1

$$\underline{\text{OPEN}} \ \underline{\text{ALL}} \ \underline{\text{AREAS}} \ \left[\underline{\text{USAGE-MODE}} \ \text{IS} \ \left[\begin{array}{l} \underline{\text{EXCLUSIVE}} \\ \underline{\text{PROTECTED}} \end{array} \right] \left\{ \begin{array}{l} \underline{\text{RETRIEVAL}} \\ \underline{\text{UPDATE}} \end{array} \right\} \right]$$

Format 2

$$\underline{\text{OPEN}} \ \underline{\text{AREA}} \, [\text{S}] \ \text{area-name-1} \ \ [,\text{area-name-2}] \ldots$$

$$\left[\underline{\text{USAGE}} \ \text{MODE} \ \text{IS} \ \left[\begin{array}{l} \underline{\text{EXCLUSIVE}} \\ \underline{\text{PROTECTED}} \end{array} \right] \left\{ \begin{array}{l} \underline{\text{RETRIEVAL}} \\ \underline{\text{UPDATE}} \end{array} \right\} \right]$$

Syntax Rules

1. Area-name-1, area-name-2, must be names which are included in the invoked subschema.

General Rules
================

1. Area-name-1, area-name-2, must be the names of areas included in the subschema invoked; otherwise, Error Status Condition 0946 results.

2. The OPEN ALL format refers to all the areas included in the subschema invoked by the run-unit.

3. Because of the file-level deadlock prevention plan of DMLCP, an OPEN can only be executed if the run-unit has no areas already open. Otherwise, Error Status Condition 0929 results. Thus, you must open any areas together that will be needed simultaneously; you are requested, additionally, to open no more areas than you will need at one time, and to close all areas during any relatively long interval when you are not using DBMS services. The CLOSE statement, unlike the OPEN, may close a subset of the areas open.

4. Use of the USAGE-MODE IS RETRIEVAL phrase (without the EXCLUSIVE or PROTECTED phrases) allows concurrent run-units to open the same area with any usage-mode other than one which is exclusive.

5. Use of the USAGE-MODE IS UPDATE phrase (without the EXCLUSIVE or PROTECTED phrases) allows concurrent run-units to open the same area with any usage-mode other than one which is exclusive or protected. See rule 8 below.

6. If the USAGE-MODE phrase is not used, then retrieval without the exclusive or protected option, called "concurrent retrieval", is assumed.

7. Use of the EXCLUSIVE phrase prevents concurrent run-units from interacting with the same area in any usage-mode.

8. Use of the PROTECTED phrase prevents concurrent update and allows concurrent retrieval within the same area if and only if before-imaging is turned off and concurrent update is not allowed by the Data Administrator. Otherwise, protected update and protected retrieval are the same as simple update and simple retrieval.

9. All specified usage-modes remain in effect until the run-unit issues a CLOSE statement for the specified areas, or until the run-unit terminates.

10. To execute a FIND, STORE, DELETE, INSERT, MODIFY, or REMOVE
    statement successfully, the run-unit must have previously opened
    (as relevant to the statement being executed):

    o   The area that contains the object record of a FIND statement,

    o   The area into which a record is to be stored.

    o   All areas containing any record occurrence which would be
        deleted or removed as a result of a DELETE statement.
        Otherwise, Error Status Condition nn01 results, where nn
        indicates the particular DML statement being attempted.

11. In addition to the areas containing the object records of the
    statements cited above, there are additional, implicit areas
    which can be impacted by DML statements. The impact can be of
    two forms:

    o   DMLCP requires information contained within the implicit
        area, in which case the area must be 'available'. For an
        area to be available, it must not be under exclusive control
        of a concurrent run-unit. It must be open for the requesting
        run-unit. If an implicit area is not available, Error Status
        Condition nn18 results (where nn indicates the particular DML
        statement attempted.)

    o   The DMLCP must alter the information contained in record
        occurrences within the implicit area, in which case the area
        must not only be available but it must permit the necessary
        alterations. Such implicit areas are deemed to be
        'affected', and must be OPEN. If an implicit area which is
        affected is not open, Error Status Condition nn21 results
        (where nn indicates the particular DML statement attempted).

12. Record occurrences which are in the search path of the sought
    record of a FIND statement, or in the search path of the sought
    record of an implicit find which occurs during the execution of a
    STORE statement, must be in areas which are open.

13. To execute an INSERT, REMOVE, STORE, DELETE, or MODIFY statement
    successfully, both the explicit and the affected implicit areas
    involved must be open with a usage-mode of update. If any of the
    areas involved are open for retrieval, Error Status Condition
    nn09 will result, (where nn indicates the particular DML
    statement attempted.)

14. Any attempt to execute an OPEN statement which would result in
    usage-mode conflict for an area will result in Error Status
    Condition 0931. The user can then choose to program a wait
    sequence or go on to do something else.

    o   The following table reflects usage-mode conflicts with an
        'N', and permitted concurrency with a 'Y'.

o   Usage-mode conflict occurs when any of the usage-mode combinations marked with an 'N' applies to an area as a result of a concurrent run-unit attempting to execute an open against it.

o   In the shaded areas (see Figure 3-1) the usage-mode conflict occurs only if before-imaging is turned off.

USAGE-MODE	OPEN-NO USAGE-MODE	RETRIEVAL	UPDATE	PROTECTED RETRIEVAL	PROTECTED UPDATE	EXCLUSIVE RETRIEVAL	EXCLUSIVE UPDATE
OPEN-NO USAGE-MODE	Y	Y	Y	Y	Y	N	N
RETRIEVAL	Y	Y	Y	Y	Y	N	N
UPDATE	Y	Y	/////	/////	/////	N	N
PROTECTED RETRIEVAL	Y	Y	/////	Y	/////	N	N
PROTECTED UPDATE	Y	Y	/////	/////	/////	N	N
EXCLUSIVE RETRIEVAL	N	N	N	N	N	N	N
EXCLUSIVE UPDATE	N	N	N	N	N	N	N

FIGURE 3-1.   USAGE-MODE RULES

15. An attempt to execute an OPEN statement on an area for which the run-unit has not satisfied the privacy locks will result in Error Status Condition 0904F and the user will be aborted.

16. If none of the Error conditions described above occurs, ERSTAT is zero, indicating successful execution of the OPEN statement.

17. If the total of areas opened by all concurrent run units exceeds an implementation limit, Error Status Condition 0905F results and the user will be aborted.

Error Status Codes for the OPEN Statement

Condition	Content of ERSTAT
Area not physically available	0928
Violation of deadlock protection rule	0929
Open blocked	0931
Specified area-name not in subschema	0946
Fatal - privacy breach attempted	-0904(0904F)
Fatal - attempted to open too many areas simultaneously	-0905(0905F)

Examples and Discussion for the OPEN Statement

Example 1

    #OPEN ALL AREAS.

This command will make available all areas which have been defined in
the subschema, for a usage- mode of simple retrieval.

Example 2

    #OPEN AREAS AREA-1, AREA-2, USAGE-MODE IS EXCLUSIVE UPDATE AREAS
        AREA-3, AREA-4 USAGE MODE IS RETRIEVAL

This command will make available AREA-1 and AREA-2 for exclusive update
and AREA-3 and AREA-4 for concurrent retrieval. This is the only way
different areas are allowed to be open for different usage-modes.   The
execution of two open commands would violate the dead lock prevention
scheme in the DBMS.

Example 3

    #OPEN ALL AREAS USAGE-MODE IS PROTECTED RETRIEVAL

This form of the open command would open all areas defined in the
subschema. The actual usage mode depends on several factors.   If
concurrent update mode and before-imaging are turned off by the Data
Administrator module, then the usage mode would be protected retrieval.

If concurrent update were allowed and before-imaging was on, then it is
possible that there would be concurrent updates but the application
program would be guaranteed a consistent view of the database for the
duration of each transaction.

```
***************
* PRIVACY KEY *
***************
```

Function

The PRIVACY KEY establishes the program and run-unit authority to execute classified DML imperative-statements in accordance with the locks declared in the schema.

General Format

Format 1

$$
\text{PRIVACY KEY [FOR}
\left\|
\begin{array}{l}
\left[\begin{array}{l} \text{EXCLUSIVE} \\ \text{PROTECTED} \end{array}\right] \text{RETRIEVAL} \\[1em]
\left[\begin{array}{l} \text{EXCLUSIVE} \\ \text{PROTECTED} \end{array}\right] \text{UPDATE}
\end{array}
\right\| \text{]OF}
$$

$$
\left\{ \begin{array}{l} \text{AREAS area-name-1 [,area-name-2]...} \\ \text{ALL AREAS} \end{array} \right\}
\text{IS}
\left\{ \begin{array}{l} \text{literal-1} \\ \text{identifier-1} \end{array} \right\}
$$

Format 2

$$
\text{PRIVACY KEY [FOR}
\left\{
\left\|
\begin{array}{ll}
\text{REST} \\
\text{STORE} \\
\text{GET} \\
\text{MODIFY} \\
\text{INSERT} \\
\text{REMOVE} \\
\text{DELETE} & \text{MANDATORY} \\
\text{DELETE} & \text{SELECTIVE} \\
\text{DELETE} & \text{ALL} \\
\text{FIND}
\end{array}
\right\|
\right\} \text{] OF}
$$

$$
\left\{ \begin{array}{l} \text{RECORDS record-name-1 [,record-name-2]...} \\ \text{ALL RECORDS} \end{array} \right\}
\text{IS}
\left\{ \begin{array}{l} \text{literal-2} \\ \text{identifier-2} \end{array} \right\}
$$

Format 3

$$
\text{PRIVACY KEY [FOR}
\left\{
\left\|
\begin{array}{l}
\text{REST} \\
\text{STORE} \\
\text{GET} \\
\text{MODIFY}
\end{array}
\right\|
\right\} \text{]OF DATA-ITEMS dbid-1 [,dbid-2]...}
$$

$$
\text{IS}
\left\{ \begin{array}{l} \text{literal-3} \\ \text{identifier-3} \end{array} \right\}
$$

Format 4

$$
\text{PRIVACY } \underline{\text{KEY}} \text{ [FOR } \left\{ \left\| \begin{array}{l} \underline{\text{REST}} \\ \underline{\text{INSERT}} \\ \underline{\text{REMOVE}} \\ \underline{\text{FIND}} \end{array} \right\| \right\} \text{] OF } \left\{ \begin{array}{l} \underline{\text{SETS}} \text{ set-name-1 } [\text{,set-name-2]}\ldots \\ \underline{\text{ALL SETS}} \end{array} \right\}
$$

$$
\underline{\text{IS}} \left| \begin{array}{l} \text{literal-4} \\ \text{identifier-4} \end{array} \right|
$$

Syntax Rules

1. Multiple PRIVACY clauses that differ only in the specification of the key are not permitted.

2. All identifiers must be defined in the FORTRAN source program.

General Rules

1. Literal-1, 2, 3, and 4 are PRIVACY KEYS: Identifier-1, 2, 3, and 4 are intended to contain PRIVACY KEYS.

2. If the optional FOR phrase is omitted, the PRIVACY KEY clause applies to all functions that could have been specified.

3. Any database-identifiers, records, sets, or areas named which are not included in the invoked subschema will be ignored.

4. If a PRIVACY LOCK has been declared in the schema, a PRIVACY command must be executed at run time to allow usage of the function even if no lock value has been assigned.

5. If the specified area-name, record-name, set-name or DBID-name is not contained in the subschema, Error Status Condition 1846 results.

Examples and Discussion for PRIVACY KEY
                            .
                            .
                            .
            DIMENSION LOCK(15)

            WRITE (1,1000)

    1000   FORMAT ('ENTER KEY FOR OPEN WITH USAGE-MODE OF UPDATE')

            READ (1,1001) LOCK

    1001   FORMAT (15A2)

            #INVOKE DBMS
                    .
                    .
                    .
                    .
            #PRIVACY KEY FOR UPDATE OF ALL AREAS IS LOCK.

            #OPEN ALL AREAS USAGE-MODE IS UPDATE.
                    .
                    .
                    .
                    .

This example illustrates the use of a PRIVACY command in a run-unit.
The   sample program reads a 30-character string from the user terminal;
it then invokes the DBMS and passes the key to unlock the open for
update.   If an improper key is passed, the open command will fail. If
the proper key is passed or the Data Administrator has not yet defined
the key value, the open will succeed.

The PRIVACY KEY command can be executed any time before the entity with
a lock defined for a particular operation is accessed in that type of
operation.

```
****************************************
* RECORD SELECTION EXPRESSIONS (rse) *
****************************************
```

Record selection expressions are used to specify the criteria by which
DBMS is to select a record in the database. The selected record
becomes the current record at the run-unit, upon which subsequent
statements may operate when accessing the database.

There are six formats of the record selection expression. These
formats permit both relative and absolute selection criteria in which
records may be selected according to their contents, their database key
value, or by currency indicators. If a currency indicator is used,
either a current record or a record related to a current record is
selected.

The record selection expression may only occur on a FIND or a FETCH
statement.

RECORD SELECTION EXPRESSIONS (rse)

Format 1

    USING identifier-1

Format 2

$$\left[\left\{\begin{array}{c} \underline{OWNER} \\ \underline{MEMBER} \end{array}\right\} \text{IN set-name- OF}\right] \underline{CURRENT} \text{ OF} \left\{\begin{array}{l} \underline{RECORD} \text{ record-name-2} \\ \underline{SET} \text{ set-name-4} \\ \underline{AREA} \text{ area-name-1} \\ \underline{RUN\text{-}UNIT} \end{array}\right\}$$

Format 3

$$\left\{\begin{array}{l} \underline{NEXT} \\ \underline{PRIOR} \\ \underline{FIRST} \\ \underline{LAST} \\ \text{integer-1} \\ \text{identifier-2} \end{array}\right\} \underline{RECORD} \text{ [record name-3] OF} \left\{\begin{array}{l} \underline{SET} \text{ set-name-5} \\ \underline{AREA} \text{ area-name-2} \end{array}\right\}$$

Format 4

    [NEXT DUPLICATE WITHIN] RECORD record-name-4

Format 5

    record-name-5 VIA [CURRENT OF] SET set-name-7 [USING dbid-3 [,dbid4]...]

Format 6

    NEXT DUPLICATE WITHIN SET set-name-8 USING dbid-5 [,dbid-6]...

Syntax Rules

Format 1:

    1.   Identifier 1 must be a three word INTEGER*2 array containing a database key.

Format 2:

    1.   Record-name-2 if specified must be defined in the subschema.

Format 3:

    1.   Integer-1 may be signed.

    2.   The data item referenced by identifier-2 must be of type INTEGER*2 and may be signed or unsigned.

    3.   If both record-name-3 and set-name-5 are used, then the record must be a member of the named set.

Format 4:

    1.   The record named by record-name-4 must have location mode CALC.

Format 5:

    1.   dbid-3, dbid-4, ... must be items defined within record-name-5.

Format 6

    1.   dbid-5, dbid-6, ... must be item defined within the record which is the current record of the set type specified in set-name-8.

General Rules

All formats :

    1.   Evaluation of a record selection expression results in the identification by the DBMS of a specific record in the database.

    2.   Records identified by a record selection expression must be stored in areas that are in an opened mode.

    3.   All records that participate in the DMLCP search for the identified record must be stored in areas that are opened.

4. Error Statuses defined in the following rules of the form mmxx may take the values 03xx if the error occurred on a FIND or 22xx if the error occurred on a FETCH.

Format 1

1. The record identified is the record whose database-key is equal to the value of the data item referenced by identifier-1.

Format 2

1. This clause without the OWNER or MEMBER phrase selects the record occurrence that is the current record of the specified record-name, set-name, or area-name, or it selects the current record of run-unit.

2. If the current record of the type specified is not known or its currency indicator is null, Error Status Condition mm06 results.

3. Use of the CURRENT OF RUN-UNIT form of the FIND permits revision of currency status indicators which were previously suppressed.

4. Set-name-3 and set-name-4 may be the same set-name or different set-names.

5. If the OWNER phrase is used, the occurrence of set-name-3 whose owner is to be selected is identified by the specified current record. Set-name-3 must not be a singular set; otherwise, Error Status Condition mm33 results.

6. If the specified current record is not of a record type defined as a member of the set in the schema, Error Status Condition mm40 results.

7. If the specified current record does not currently participate as a member of any occurrence of set-name-3, and if it is not known as the current record of set-name-3 by the run-unit, Error Status Condition mm22 results.

8. If the MEMBER phrase is used, the record selected is the first member (in terms of logical order of the set) in the set occurrence of set-name-3 owned by the specified current record.

9. If the specified current record is not of the record type defined to be the owner record type of set-name-3 in the schema, Error Status Condition mm41 results.

10. If the set occurrence of set-name-3 owned by the specified current record is null, Error Status Condition mm26 results.

Format 3

1. In Format 3, if the record-name-3 phrase is used, only occurrences of record-name-3 will be considered in evaluating the record selection expression. If area-name-2 is stated, then the WITHIN clause for record-name-3 must include area-name-2; otherwise, Error Status Condition mm42 results. If set-name-5 is stated, record-name-3 must be defined as a member of set-name-5; otherwise, Error Status Condition mm40 results.

2. If a set-name is specified, the set occurrence from which the object record is to be selected is identified by the current record of the specified set. If the current record of that set is not known or its currency indicator is null, Error Status Condition mm06 results.

3. If an area-name is specified, the object record is selected from the named area. If the named area is not open, Error Status Condition mm01 results.

4. If the NEXT or PRIOR phrase is used and an area-name is specified, and the current record of the named area is not known or its currency indicator is null, Error Status Condition 0306 results.

5. NEXT RECORD OF area-name AREA means the record physically closest in the "next" direction to the current record of the named area. If there is no such record in the area named, Error Status Condition mm07 results.

6. PRIOR RECORD OF area-name AREA means the record physically closest in the "prior" direction to the current record of the named area. If there is no such record in the area named, Error Status Condition mm07 results.

7. NEXT RECORD OF set-name SET means the subsequent record relative to the current record of the named set in the logical order of the set regardless of the database-key sequence. If the set is empty, that is, no member record occurrences participate in the set, Error Status Condition mm26 results. If the current record is the last record in the set, Error Status Condition 0307 results.

8. PRIOR RECORD OF set-name SET means the previous record relative to the current record of the named set in the logical order of the set regardless of database-key sequence. If the set is empty, that is, no member record occurrences participate in the set, Error Status Condition mm26 results. If the current record is the first record in the set, Error Status Condition 0307 results.

9. FIRST RECORD OF area-name AREA is the record occurrence physically first in the named area. If there are no records in the named area, Error Status Condition mm26 results.

10. LAST RECORD OF area-name AREA is the record occurrence physically last in the named area. If there are no records in the named area, Error Status Condition mm26 results.

11. FIRST RECORD OF set-name SET is the first member occurrence in terms of the logical order of the set. The record selected is the same as would be selected if the current record of the set was the owner record and the NEXT RECORD OF set-name SET was used. If the set occurrence is empty, that is, no member record occurrences participate in the set, Error Status Condition mm26 results.

12. LAST RECORD OF set-name SET is the last member record occurrence in terms of the logical order of the set. The record selected is the same as would be selected if the current record of the set was the owner record and the PRIOR RECORD OF set-name SET was used. If the set occurrence is empty, that is, no member record occurrences participate in the set, Error Status Condition 0326 results.

13. Identifier-2 must be initialized with an integer prior to execution of the FIND statement. Identifier-2 and integer-1 represent the ordinal count of the object record occurrence relative to the beginning, if positive, or ending, if negative of a set occurrence or area. A negative value selects in the prior direction and a positive value in the next direction of the set occurrence or area.

14. If there are no records in the set occurrence or area named, Error Status Condition 0326 results. If the value of integer-1 or the contents of identifier-2 are greater than the number of record occurrences in the set occurrence or area specified, then Error Status Condition mm07 results.

Format 4

1. In Format 4, the LOCATION MODE IS CALC clause must have been used in the description of record-name-4; otherwise, Error Status Condition mm45 results. All search arguments specified in the CALC clause must be included in the invoked subschema; otherwise, Error Status Condition mm43 results.

2. Prior to the execution of the FIND statement, the data items specified in the LOCATION MODE clause must be initialized and if the related WITHIN clause includes more than one area, the data item declared as AREA-ID must also be initialized with the area-name to which the resulting database-key applies. If an area-name is specified which is not included in the WITHIN clause, Error Status Condition mm23 occurs. If no record with

the specified CALC key values is found, Error Status Condition mm26 results.

3. When the NEXT DUPLICATE phrase is not used, the first record found by DMLCP which satisfies the argument values for the CALC keys in User Work Area is the one selected. When the NEXT DUPLICATE phrase is used and the following conditions exist, the record occurrence selected will be the next record found by DMLCP with the same value for its CALC key as the current record of the run-unit. If no such duplicate record occurrence is found, then Error Status Condition mm26 results.

   o The CALC key in UWA is equal to the value of the CALC key in the current record of run-unit.

   o The implied area-name or the area-name explicitly specified in AREA-ID is the area in which the current record of run-unit exists.

4. When the NEXT DUPLICATE phrase is used and the above conditions do not exist, the effect is the same as if the NEXT DUPLICATE phrase had not been stated.

Format 5

1. Record-name-5 must be defined in the schema as a member of set-name-7; otherwise, Error Status Condition 0340 results.

2. Format 5 of the rse, when the CURRENT phrase is used, causes record selection based on the current set occurrence of set-name-7. When the CURRENT phrase is not used it causes record selection based on the SET OCCURRENCE SELECTION clause defined for the named record and set. All search arguments specified in that clause must be included in the invoked subschema; otherwise, Error Status Condition mm44 results. The search arguments must be initialized prior to the execution of the FIND statement. In addition, if the process involves selection of any record on the basis of a LOCATION MODE IS CALC clause, any AREA-ID in the WITHIN clause for that record must be initialized. If AREA-ID is not initialized with an area-name specified or implied for the record type, Error Status Condition mm02 results. If the database-key developed from the CALC keys is inconsistent with the area-name specified, Error Status Condition 0302 results.

3. The record selected if the USING phrase is not specified is the first record occurrence of the record type specified by record-name-5, which is in the selected occurrence of the named set. If the USING phrase is employed, data-base-identifier-3 and data-base-identifier-4 are implicitly qualified by record-name-5 and must be the names of fixed length data items included in record-name-5; otherwise, Error Status Condition mm04 results. DMLCP will select a record having the same values for

data-base-identifier-3 and data-base-identifier-4 as are in the
User Work Area. If more than one record occurrence meets the
criteria specified, the first such record occurrence will be
selected in terms of scanning the set in the next direction.   If
no record is found which meets these criteria, Error Status
Condition mm26 results. When the CURRENT phrase is used, if the
current record of set-name-7 is not known or its currency
indicator is null, Error Status Condition mm06 results.

Format 6

1. Format 6 of the rse causes a search of the members of the current
    set occurrence of set-name-8 for a record which is of the same
    type as the current record of set-name-8, and, which has the same
    values for data-base-identifier-5, data-base-identifier-6. All
    values in the User Work Area are ignored. The search is in the
    next direction and starts from the current record of set-name-8.
    It continues until either a duplicate is found or the end of set
    is reached. If no duplicate is found, Error Status Condition
    0307 results. If the current record of set-name-8 is not known
    or its currency indicator is null, Error Status Condition mm06
    results. If the current record of set-name-8 has been deleted by
    this or another run-unit, since it became current in this
    run-unit, Error Status Condition mm17 results.   If
    data-base-identifier-5, data-base-identifier-6 are not defined in
    the subschema invoked as part of the current record of
    set-name-8, Error Status Condition mm04 results.

Examples and Discussion of the RECORD SELECTION EXPRESSIONS

For the following examples assuming three is a database with these record types: A DEPARTMENT (DEPT) record, an EMPLOYEE record and a SUPERVISOR record. The DEPT record has a location mode of CALC on the DEPTNM field and is the owner of the DEPARTMENT-SET (DEPT-SET) of which the EMPLOYEE and the SUPERVISOR records are a member. The set is sorted on an EMPLOYEE NUMBER field which is part of both the EMPLOYEE record and the SUPERVISOR record. The database has been pre-stored with the following set occurrence.

(DEPT-SET)
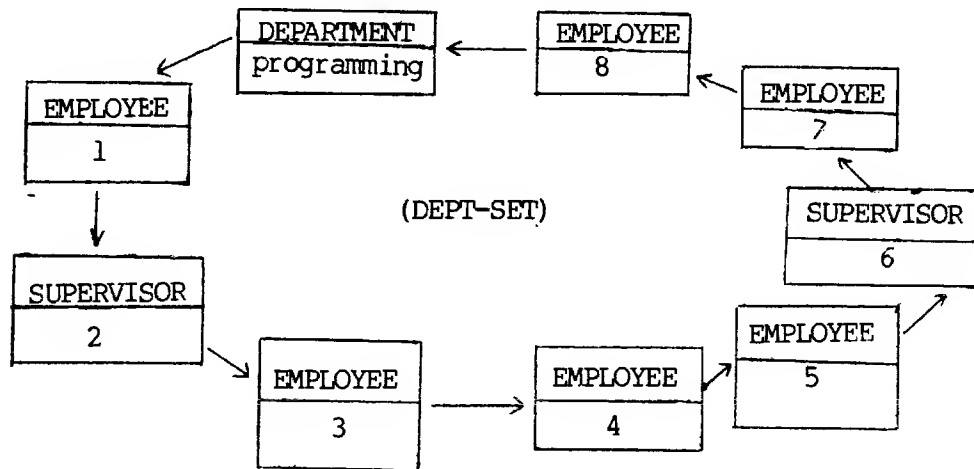
Example 1

#FIND USING DBK1.

The FORTRAN identifier DBK1 has been declared to be a 3-element INTEGER*2 ARRAY which has been initialized to a valid database-key. The result of this command is that the record specified by this DBK becomes current at the run-unit area in which it is stored, of record and of all sets in which it participates as an owner or member.

Example 2

#FIND OWNER IN DEPT-SET OF CURRENT OF RECORD EMPLOYEE.

Assuming that the current of RECORD Employee is EMPLOYEE 4, the owner record DEPARTMENT which is programming will be selected and become current at run-unit, current at record type DEPARTMENT, current at set DEPARTMENT-SET and current at the area in which the DEPT record resides.

Example 3

    #FIND -2 RECORD SUPERVISOR OF SET DEPT-SET.

    The format of the rse will traverse the DEPT-SET in the prior
    direction and produce the supervisor record with the number 2 as
    the current of run-unit at all. It should be noted that the
    employee record in the set did not affect the count. Had the
    "RECORD SUPERVISOR" clause been dropped from the FIND, the Employee
    Record 8 would be the record selected.

Example 4

    #FIND NEXT RECORD EMPLOYEE OF SET DEPT-SET.

    Assuming that the current record of run-unit is Employee 1, the
    record chosen by this format of the record selection expression
    would be Employee 3. This record would then become current of
    run-unit, etc.

Example 5

    #FIND RECORD DEPT.

    Assuming the Department name field was assigned the value of the
    Dept name, the Department record (whose field matched the Dept
    name) would have been selected and would have become current of
    run-unit.

Example 6

    #FIND EMPLOYEE VIA DEPT-SET USING EMPNUM.

    Assuming the set selection clause for Dept Set is location mode of
    owner and the call field for the Dept record was assigned to
    PROGRAMMING and the database-identifier EMPNUM was assigned the
    value 5, this format at the record selection expression would have
    first located the record Dept which had the DEPTNUM field with the
    value programming and then located the Employee record with the
    EMPNUM of 6, which was owned by that Department. The EMPLOYEE
    record would then become current of run-unit, current of record
    EMPLOYEE, current of set DEPT-SET, and current of area in which it
    is stored.

```
**********
* REMOVE *
**********
```

Function

To cancel the membership of the object record in the occurrences of the specified set-names in which it currently participates as a member, provided that the object record is defined as an optional member of the sets named.

General Format

$$\text{\underline{REMOVE FROM}} \left\{ \begin{array}{l} \underline{\text{SETS}} \text{ set-name-1 } [,\text{set-name-2}]\dots \\ \underline{\text{ALL SETS}} \end{array} \right\}$$

Syntax Rules

1. The sets set-name-1, set-name-2 must be defined in the invoked subschema.

2. The record which is current at run-unit must be defined as an optional member of the named sets in the Schema DDL.

General Rules

1. The object record of the REMOVE statement is the current record of the run-unit.

2. If set-names are specified in the REMOVE statement, then the object record must have been defined in the schema as an optional member of all of the sets named. It must also currently participate as a member in an occurrence of each of the sets named.

3. The ALL SETS format is designed to remove the object record from all set occurrences in which it is defined in the invoked subschema as an optional member and in which it currently participates as a member.

4. No change occurs to any currency information maintained by the
    DMLCP.

5. If the run-unit has not satisfied the privacy locks for all
    records, sets, areas, and data items needed to execute the REMOVE
    statement, Error Status Condition 1104F will result and the
    run-unit will be aborted by DMLCP.

6. In addition to the conditions described, if any of the following
    conditions are encountered, the REMOVE statement is not
    successfully executed, the database remains in the state existing
    prior to the attempted execution, and the appropriate Error
    Status Condition code is made available in special register
    ERSTAT. Otherwise, the contents of ERSTAT is set to zero,
    indicating successful execution of the REMOVE statement.

    o   If all set-names specified are not included in the invoked
         subschema, Error Status Condition 1146 results.

    o   If the current record of the run-unit is not known, Error
         Status Condition 1113 results.

    o   If the object record is not defined as an OPTIONAL MEMBER of
         all of the specified set-names, Error Status Condition 1115
         results.

    o   If the object record does not currently participate as a
         member in an occurrence of at least one of the sets specified
         or implied, Error Status Condition 1122 results.

    o   If the object record or any record occurrence affected by the
         REMOVE statement is located in an area which is open for
         RETRIEVAL, Error Status Condition 1109 results.

    o   If any record occurrence needed by the DMLCP for
         informational purposes (such as following a search path) is
         not available because it is off-line or under exclusive
         control of another run-unit, Error Status Condition 1118
         results.

Error Status Codes for REMOVE

Condition	Content of ERROR-STATUS
Incorrect usage-mode for area	1109
No current record of run-unit	1113
Object record not defined as an optional member of a named set	1115
Implicitly referenced area not available	1118
Object record not currently a member of named or implied set	1122
Concurrent update conflict	1135
Specified set-name not in subschema	1146
Fatal - privacy breach attempted	-1104(1104F)

Examples and Discussion for REMOVE

    #REMOVE FROM ALL SETS.

This format of the REMOVE statement will take the current record of
run-unit and remove it from all sets of which it is an optional member.
That is, the record will no longer be accessible from the set
occurrences from which it has been removed. It will still be
accessible from other set occurrences of which it is a mandatory member
or from its CALC key if the record has a location mode of CALL or from
a traversed of the AREA.

```
*************************
* START OF TRANSACTION *
*************************
```

Function

The START TRANSACTION command defines the start of either an update or
retrieval activity on the database. It provides a retrieval
transaction and a consistant view of the database. It allows an update
transaction to define the beginning of a recoverable series of commands
and it guarantees exclusive update rights to date, which may be
modified. It also saves the currency indicators as they are defined at
the beginning of transaction.

General Format

$$\text{START TRANSACTION identifier} \quad \left[\begin{array}{l} \underline{\text{UPDATE}} \\ \text{'} \underline{\text{RETRIEVAL}} \end{array} \right]$$

Syntax Rules

1. ID is a FORTRAN identifier which contains an INTEGER*2 value
    assigned to the transaction as the users (run-units) name for
    that transaction.

General Rules

1. All commands which access the database (FIND, GET, FETCH, MODIFY,
    STORE, INSERT, REMOVE, DELETE, IF, MOVE) must be nested between
    START TRANSACTION and END OR ABORT TRANSACTION statements.

2. The following commands may not be nested between the START/END
    TRANSACTION statement: INVOKE, OPEN, CLOSE, START TRANSACTION,
    and EXIT DBMS.

3. If Before-imaging is turned off by the DBACP, roll back cannot
    occur.

4. If a subsequent update command (MODIFY, STORE, INSERT, REMOVE,
    DELETE) attempts to modify data that has been modified by another
    update transaction that is currently active, the update command
    will receive an Error Status of mm35 where mm is the major code.
    The only choice allowed the application at that point is to abort
    the transaction and start it over again. If this is not done on
    execution of the next DML command, the transaction will be rolled
    back and the run-unit aborted.

5. If a START TRANSACTION command is executed when a transaction is
    already active, Error Status Condition 2335 results.

Example and Discussion of START OF TRANSACTION

```
        INTEGER TRANID
              .
              .
              .
        #INVOKE DBMS

        #OPEN ALL AREAS USAGE-MODE IS UPDATE

        TRANID=1


  5     #START TRANSACTION TRANID, UPDATE

        #FIND RECORD DEPT

        #STORE EMPLOYEE

        IF (ERSTAT.EQ.0) GO TO 10

        #CLEAR ERROR

        #ABORT TRANSACTION TRANID

        GO TO 5
 10     CONTINUE
              .
              .
              .
        #END TRANSACTION TRANID
              .
              .
              .
        GO TO 5
```

The above program is an example of a procedure that might be used to add an employee record to a department in a concurrent on-line environment.

The DBMS is first invoked and all areas defined in the subschema are open. Then, a start of transaction is executed. During the transaction a FIND statement is executed and then a STORE. If the STORE statement has a concurrent update error, the transaction is rolled back and tried again. If it is successful, execution continues until the END TRANSACTION, which then secures all updates to disk and defines the start of a quiescent point until the next START TRANSACTION.

```
********
* STORE *
********
```

Function

The STORE statement accomplishes the following:

1. Acquires space and a database-key for a new record occurrence in the database.

2. Causes the values of the appropriate data items in User Work Area to be included in the occurrence of the object record in the database.

3. Inserts the object record into all sets for which it is defined as an automatic member in the schema.

4. Establishes a new set occurrence for each set where the object record is defined as owner in the schema.

5. Establishes the object record as the current record of the run-unit.

6. Depending on the SUPPRESS status, establishes the object as:

   o   The current record of the area in which it is stored,

   o   The current record of the record-name,

   o   The current record of set for all set-names in which it is specified as an owner or automatic member.

General Format

STORE record-name

Syntax Rules

1. The invoked subschema must include the named record; the data items or set specified in the LOCATION MODE clause of the named record; at least one of the areas specified in the WITHIN clause of the named record; all sets in which the named record is defined as an owner or an automatic member; all data items, records, and sets specified or referenced in the SET OCCURRENCE SELECTION clauses and ASCENDING/DESCENDING KEY clauses of those sets in which the named record is defined as an automatic member.

General Rules

1. The subschema invoked must include the named record;   otherwise, Error Status Condition 1246 results.

2. The data items, data-base-data-names, or sets specified in the LOCATION MODE clause of the named record must be included in the invoked subschema;   otherwise, Error Status   Condition   1243 results.

3. The area in which the record is to be stored must be included in the invoked subschema;   otherwise, Error Status Condition 1247 results.

4. All sets in which the named record is defined as an automatic member must be included in the invoked subschema; otherwise, Error Status Condition 1208 results.

5. All data items, data-base-data-name, records, and sets specified or referenced in the SET OCCURRENCE SELECTION clauses of those sets in which the named record is defined as an automatic member must be included in the subschema;   otherwise, Error Status Condition 1244 results.

6. All data items specified in the ASCENDING/DESCENDING KEY clauses of those sets in which the named record is defined as an automatic member must be included in the invoked subschema; otherwise, Error Status Condition 1248 results.

7. A database-key and space for the object record are allocated on the basis of the description of the record in the subschema invoked by the run-unit and values provided by the user.

    o   Data items are included in the object record of the STORE statement   from the User Work Area. Data   items   not initialized will be given indeterminate values.

    o   Also included is any control data item, as specified or referenced in the SET OCCURRENCE SELECTION clauses for this record, when all sets are defined as automatic members.   If any such SET OCCURRENCE SELECTION clause is THRU CURRENT SET, the user must insure that the current record of that set identifies the proper set occurrence.   If any such SET OCCURRENCE SELECTION clause is THRU LOCATION MODE OF OWNER, then all control data items specified for that owner record must be initialized.   In addition, all other data items specified in the applicable SET OCCURRENCE SELECTION clauses must be initialized.

8. For each LOCATION MODE clause involved in the execution of a STORE, the following control data items (depending on the option used) must be initialized.

o   In the case of the DIRECT phrase, the data item specified in
    the   LOCATION MODE clause   must be initialized. In a STORE
    statement, the first 28-bits of such a data item is ignored.
    The   third word of the integer array must be initialized with
    zero or a positive integer number. If a non-zero value is
    given,   this word will be used by the DBMS   as   the
    Bucket-number portion of the database-key. If a zero has
    been used, DBMS will assign an available Bucket number for
    the record type in the specified area. In either case, the
    DMLCP will generate the first 20-bits of the database-key
    (which represents the record type and the area in which the
    occurrence is to be stored).   The system will assign the
    resulting 48-bit DBK, if it is available, to the record which
    is the object of the STORE statement.   If   the
    occurrence-number portion was supplied by the user, and the
    resulting database-key is not available, DMLCP will assign an
    occurrence-number which will result in an available DBK in
    the   specified area.   In any case, the   full   48-bit
    database-key by which the record occurrence is stored will be
    returned in UWA in the data item specified in the DIRECT
    phrase of the LOCATION MODE clause.

A database-key can be determined by the DMLCP to be 'not
available' for either of the following reasons:

o   The database-key is already in use either by a current or
    a deleted record occurrence of the database.

o   The storage structure implemented under the   current
    version of the DMLCP will not allow the database-key,
    since it will cause overflow.

    If the DIRECT phrase is used, and the WITHIN clause is
    specified, the data item declared as AREA-ID must be
    initialized with an area-name specified or implied for
    the record type or to zero, in which case the first area
    in which the record definition is included in the schema
    will be chosen.   If an area-name is specified which is
    not defined in the schema as a possible area for this
    record type, Error Status Condition 1223 results.

    o   In the case of the CALC phrase, the
        database-identifiers   named in the LOCATION MODE
        clause must be initialized, and if the record type
        can be stored in more than one area, the data item
        declared as AREA-ID in the WITHIN clause must be
        initialized with an area-name specified or implied
        for the record type (or zero, as above).   If an
        area-name is specified which is not defined in the
        schema as a possible area for the record type, Error
        Status Condition 1223 results. The DMLCP will
        develop a database-key which is compatible with the
        specified area-name.

o   In the case of the VIA set-name phrase, the data
    items specified in the SET OCCURRENCE SELECTION
    clause for this record in the named set must be
    initialized, and, if the record type can be stored in
    more than one area, the data item declared as AREA-ID
    in the WITHIN clause must be initialized with an
    area-name specified or implied for the record type
    (or zero, as above). Initialization of the data
    items specified in the SET OCCURRENCE SELECTION
    clause is required to enable the DMLCP to select a
    unique occurrence of the named set, and is required
    regardless of whether the record being stored is an
    automatic or manual member of that set. If it is an
    automatic member of the set, it will be logically
    inserted into the selected set occurrence. If it is
    a manual member, it will not be inserted into the
    selected set occurrence. In both cases, however,
    subject to the constraints of the implied area and
    areas specified in the WITHIN clause, the record
    being stored will be placed by the DMLCP as close as
    is possible to the actual or probable logical insert
    point in the selected set occurrence. The effect of
    implicit and explicit areas on placement is as
    follows: if the record type is defined in the schema
    to be a possible occurrence of only one area-name,
    placement is controlled by that area-name; If more
    than one area-name is specified, the initialized
    value of AREA-ID controls placement. AREA-ID must be
    initialized with an area-name appropriate for the
    record type or it must be initialized with a zero;
    otherwise, Error Status Condition 1223 results. If
    AREA-ID is initialized with a legal area-name,
    placement is in that area. If AREA-ID is initialized
    with a zero, placement occurs as close as is possible
    to the logical insert point. Such placement is,
    however, constrained by the implied or specified
    areas for this record.

9. If no LOCATION MODE clause has been specified for the record
   being stored, DBMS will assign a database-key consistent with the
   relevant area-name. If the record can be stored in more than one
   area, AREA-ID specified in the WITHIN clause must be initialized
   with an area-name specified or implied for the record type or
   zero (as above). If an area-name is specified which is not
   included for the record type, Error Status Condition 1223
   results.

10. In cases where the type of a data item, as defined in the
    subschema invoked by the run-unit, differs from the definition of
    that data item in the schema:

    o   Conversions performed by the DMLCP will be in accordance with
        the rules specified in the TYPE clause of the DDL for the

SCHEMA and the TYPE specification of the subschema DDL for FORTRAN.

11. Data items defined in the schema for the database but not included in the subschema invoked by the run-unit are not assigned User Work Area locations. Null-values will be placed in the database for such data items.

12. The object record occurrence is inserted into a set occurrence for each set in which the record is defined as an automatic member. The ordering rules for the set govern the insertion point of the object record in all of the relevant set occurrences.

13. The object record is established as the owner of a set occurrence for each set in which it has been defined as an owner. These set occurrences are empty at this time; that is, they have no member records.

14. The successfully stored record occurrence becomes the current record of the run-unit.

15. If suppression of currency updates is not in effect (see SUPPRESS command), the object record also becomes the current record of the area in which it is placed, the current record of its record-name, and the current record of all set-names in which it is defined as an owner or automatic member.

16. The SUPPRESS command provides the selective facility to prevent the object record from becoming the current record of the area, the current record of its record-name, and the current record of any or all of the set-names in which it is defined as either an owner or an automatic member. When SUPPRESS is used, the specified currency indicators are not affected by the execution of the STORE statement.

Use of SUPPRESS cannot prevent the object record from becoming the current record of the run-unit.

17. If any of the following conditions are encountered, the STORE statement is not successfully executed, the database and User Work Area remain in the state existing prior to the attempted execution, and the appropriate Error Status Condition code is made available thru the special register ERSTAT. Otherwise, ERSTAT will be zero, indicating successful execution of the STORE statement.

   o   If the named record is not included in the invoked subschema, Error Status Condition 1246 results.

   o   If all database entities specified in the LOCATION MODE clause of the named record are not included in the invoked subschema, Error Status Condition 1243 results.

o   If the area in which the named record is to be stored is not
     included in the invoked subschema, Error Status Condition
     1247 results.

o   If the object record is to be stored in an area which is not
     open, Error Status Condition 1201 results.

o   If the object record or any record occurrence affected by the
     STORE statement is located in an area which is only open for
     retrieval, Error Status Condition 1209 results.

o   If any record occurrence needed by the DMLCP for
     informational purposes (such as following a search path) is
     not available because it is off-line or under the exclusive
     control of another run-unit, Error Status Condition 1218
     results.

o   If media space is not available, (i.e., there is insufficient
     room in the area specified or in a set where insertion would
     occur for the new record occurrence), Error Status Condition
     1211 results.

o   If the area specified for the object record, or for any
     record selected through its CALC keys as a result of the
     execution of the STORE statement, is not one of those
     implicitly or explicitly specified in the schema definition
     of the relevant record type, Error Status Condition 1223
     results.

o   If the record to be stored would violate a DUPLICATES NOT
     ALLOWED clause defined for any of the records or sets
     involved, Error Status Condition 1205 results.

o   If all sets in which the named record is defined as an
     automatic member are not included in the invoked subschema,
     Error Status Condition 1208 results.

o   If for any of the set-names involved, a set occurrence which
     meets the relevant set selection criterion cannot be found,
  , Error Status Condition 1225 results.

o   Error Status Condition 1244 results if all database entities
     specified or referenced in the SET OCCURRENCE SELECTION
     clauses of those sets in which the named record is defined as
     an automatic member are not included in the invoked
     subschema.

    o   If all data items specified in the ASCENDING/DESCENDING KEY clauses or search key clauses of those sets in which the named record is defined as an automatic member are not included in the invoked subschema, Error Status Condition 1248 results.

    o   If a CHECK clause applies and any of the stored data items is detected as invalid, Error Status Condition 1227 results.

    o   If the value of the data item in the User Work Area is such that it cannot be converted to the format specified in the schema for that data item, Error Status Condition 1219 occurs.

18. If the run-unit has not satisfied the privacy locks on all data items, records, sets, and areas needed to execute the STORE, Error Status Condition 1204F will result and the run-unit will be aborted by the DMLCP.

Error Status Codes for the STORE Statement

Condition	Content of ERROR STATUS
Area not open	1201
Violation of DUPLICATES NOT ALLOWED clause	1205
Referenced set-name not in subschema	1208
Incorrect usage-mode for area	1209
Media space not available	1211
Implicitly referenced area not available	1218
Conversion of value of data item not possible	1219
Illegal area-name	1223
No set occurrence satisfies argument values specified	1225
Concurrent update error	1235
New value of data item or actual result data item violates CHECK clause	1227
Arguments of Location Mode clause not included in subschema	1243
Arguments of Set Occurrence Selection clause not included in subschema	1244
Specified data item, record, set, or area not in subschema	1246
Referenced area-name not in subschema	1247
Arguments of ASCENDING/DESCENDING KEY clause not in subschema	1248
Fatal - privacy breach attempted	-1204(1204F)

Examples and Discussion for the STORE Statement

        #STORE DEPT

The STORE command creates the record in the database, assigns it a DBK, inserts it into all sets for which it is an automatic member, and creates entries for all sets which it owns. Data items which are declared in the schema but not defined in the subschema are filled with null values.

```
*************
* SUBSCHEMA *
*************
```

Function

To name the subschema which will provide the description of the data to the application program.

General Format

    SUBSCHEMA subschema-name of SCHEMA schema-name

Syntax Rules

1. The schema-name specified must be the name of a schema known to the DBMS.

2. The subschema-name specified must refer to a FORTRAN subschema for the named schema and must be known to the system.

3. This statement must be present in any program unit containing FDML commands. It must precede all executable FORTRAN statements.

4. If there is more than one program unit in a file, the SUBSCHEMA statements in those program units must all refer to the same schema-name and subschema-name. This error cannot be detected by the DBMS and must therefore be verified by the user.

General Rules

None.

Example and Discussion of the SUBSCHEMA Statement

```
#SUBSCHEMA MY-SUB OF SCHEMA THE-SCHEMA.
INTEGER I,J,K
end
SUBROUTINE SUB1
#SUBSCHEMA MY-SUB OF SCHEMA THE-SCHEMA.
INTEGER I,J,K
            .
            .
            .
end
SUBROUTINE SUB 2
#DBMS SUBPROGRAM
INTEGER I,J,K
            .
            .
            .
END
```

The above example shows how a sample application program might appear in a file. This program consists of a main routine and two subroutines which will to reference the database.

If more than one SUBSCHEMA statement is used, then it must refer to the same schema and subschema. The DBMS SUBPROGRAM statement may be used as shown in the third program unit in place of the subschema statement if a previous subschema statement has been declared.

All of the above statements are replaced by the common block declarations which serve as the User Work Area.

```
************
* SUPPRESS *
************
```

Function

Suppresses, that is prevents, the updating of any combination of the currency indicators: Current record of areas, current record of records, and current record of sets, or optionally, current record of specific sets.

Optionally, it clears all suppression which exists and indicates new suppression of currency indicators.

General Format

$$
[\text{CLEAR}] \left[\underline{\text{SUPPRESS}} \left\{ \left\| \begin{array}{l} \underline{\text{ALL}} \\ \underline{\text{RECORD}} \\ \underline{\text{AREA}} \\ \left\{ \underline{\text{SET}} \atop \text{set-name-1 [set-name-2]} \ldots \right\} \end{array} \right\| \right\} \right]
$$

Syntax Rules

1. The sets set-name-1, set-name-2, must be defined in the invoked subschema.

General Rules

1. The SUPPRESS statement will dictate the currency indicator updates performed by succeeding FIND, FETCH, STORE, or INSERT commands.

2. If the CLEAR option is not used, all currency indicators specified in the SUPPRESS statement are in addition to any currency indicators specified for suppression previously.

3. If the CLEAR option is used, the suppression now existing on any currency indicators will be lifted. Then, if the SUPPRESS phrase is also used, the specified currency indicators will be set for suppression of currency updates.

4. If all set-names specified are not included in the invoked subschema, Error Status Condition 1546 results; otherwise, special register ERSTAT is set zero, indicating successful execution of the SUPPRESS statement.

Error Status Condition of the SUPPRESS Statement

Condition	Content of ERSTAT
Set-name specified not in subschema	1546

Example and Discussion of the SUPPRESS Statement

.
.
.
.

#FIND FIRST RECORD EMPLOYEE OF SET DEPT-SET.

.
.

#SUPPRESS SET CURRENCY UPDATES.

.
.

#FIND NEXT RECORD EMPLOYEE OF SET DEPT-SET.

#CLEAR.

#FIND NEXT RECORD EMPLOYEE OF SET DEPT-SET.

#FIND NEXT RECORD EMPLOYEE OF SET DEPT-SET.

This program is an example of the use of the SUPPRESS statement. The
first FIND makes the object record current of AREA, RECORD, and all
sets which it owns or of which it is currently a member. The
subsequent SUPPRESS command will turn off all currency updates for any
set. The next FIND will update current of run-unit, current of area,
and current of record, but will not update any set currency pointers.
Thus, any reference to the current of set by an INSERT or FIND command
will reference the set currency indicators defined by the first FIND.

In the example, the object record of the third FIND would be the same
as the object record of the second FIND because the current of set
DEPT-SET was not updated. After the CLEAR command is executed, the
object record of the fourth FIND would be the third record of the set
because the third FIND updated the currency pointer of that set.

SECTION 4

FORTRAN DDL & DML DIAGNOSTIC METHODS

SUMMARY OF MAJOR CODES

Major Code	DML Statement
01	CLOSE
02	DELETE
03	FIND
05	GET
07	INSERT
08	MODIFY
09	OPEN
10	CLEAR ERROR EXIT DBMS
11	REMOVE
12	STORE
13	PRIVACY KEY
14	INVOKE
15	SUPPRESS
18	IF
19	MOVE
22	FETCH
24	START TRANSACTION
25	END TRANSACTION
26	ABORT TRANSACTION

NOTE

ERROR CODE FORMAT
is XXYY where
major code is XX
and error code is YY

SUMMARY OF NON-FATAL MINOR CODES

Minor Code	Error Condition
01	Area not open
02	Database-key inconsistent with area-name
03	Invoke has already been executed
04	Data items invalid or inconsistent
05	Violation of DUPLICATES NOT ALLOWED clause
06	Current of set, area, or record-name not known
07	End of set or area
08	Referenced record or set-name not in subschema
09	Incorrect usage-mode for area
10	Not used
11	Media space not available
12	Database-key not available
13	No current record of run-unit
14	Object record is mandatory automatic in named set
15	Object record is mandatory in named set
16	Record already a member of named set
17	Deleted record involved
18	Implicitly referenced area not available
19	Conversion of value of data item not possible
20	Not used
21	Not used, no current owner record of named set
22	Record not currently a member of named set
23	Illegal area-name
24	Not used

25   No set occurrence satisfies argument values

26   No record satisfies the FIND specified

27   CHECK clause violation

28   Not used

29   Violation of deadlock protection rule

30   Unqualified delete attempted on non-empty set

31   OPEN blocked

32   Deleted set occurrence involved

33   Attempted to find owner of a singular set

34   Time out

35   Concurrent update conflict, refer to 35A

36   Not used

37   Not used

38   Not used

39   Not used

40   Record type not a member of named set

41   Record type not owner of named set

42   Record type not included in named area

43   Arguments of LOCATION MODE clause not included in subschema

44   Arguments of SET OCCURRENCE SELECTION clause not included in
        subschema

45   Location mode of record not specified as CALC

46   Specified data item, record, set or area not in subschema

47   Referenced area-name not included in subschema

48   Arguments of ASCENDING/DESCENDING clause not in subschema

49   Arguments of within clause not included in subschema

50   Not used

51	Not used
52	Virtual data item not available, not used
53	Invalid value for virtual result data item, not used
54	Value of string data item truncated in User Work Area
01F	Required subschema has been deleted
02F	Required schema has been deleted
03F	Not used
04F	Privacy breach attempted
05F	Attempted to open too many areas simultaneously
06F	Not used
07F	Not used
08F	Not used
09F	Feature not implemented
10F	Internal fatal error
11F	Schema locked
12F	Database files not allocated
13F	Volume not on system
14F	Too many files opened on schemas in concurrent use
15F	User Work Area larger than declared space in dynamic invoke

NOTE
Fatal errors are indicated by
a negative number. (e.g., - 0311
is a fatal error for the FIND
command).

SUMMARY OF CONCURRENT ACCESS CONFLICTS

Message	Meaning
1) ATTEMPTED ACCESS TO CONCURRENTLY UPDATED BLOCK	External conditions
2) ATTEMPTED ACCESS TO INVALID BEFORE-IMAGE	Slow reader – requested before image has been recycled
3) TRANSACTION MUST ABORT	A concurrency error has already occurred.
4) NO TRANSACTION IS IN EFFECT	DML program error.
5) OVERFLOW OF BIT MAP	Too many updates or a slow update or an aborted update.
6) OVERFLOW OF OLDEST BEFORE-IMAGE TABLE	Same as above.
7) OVERFLOW OF GENERATION NUMBER	More than 232 transactions (update transactions).
8) OVERFLOW OF BEFORE-IMAGE FILE	Too much update activity.
9) ATTEMPTED WRITE BY READ TRANSACTION	DML program error.
10) INVALID START UPDATE WITHIN ACTIVE TRANSACTION	DML program error.
11) INVALID START RETRIEVAL WITHIN ACTIVE TRANSACTION	DML program error.
12) INVALID ABORT TRANSACTION -- NO TRANSACTION ACTIVE	DML program error.
13) INVALID END TRANSACTION -- NO TRANSACTION ACTIVE	DML program error.
14) OPEN IS INVALID WITHIN ACTIVE TRANSACTION	DML program error.
15) CLOSE OR EXIT DBMS IS INVALID WITHIN ACTIVE TRANSACTION	DML program error.
16) CANNOT READ OR WRITE BEFORE, LOG OR AFTER FILE	DBMS error.
17) LOG FILE NOT OPEN	Can't happen currently.

18) OVERFLOW OF READ TRANSACTION   More than 232 read transactions.
        NUMBER

19) BEFORE-IMAGE FILE NOT OPEN   DBA error.

20) INCONSISTENT TRANSACTION   Identifier on END or ABORT
        IDENTIFIER   TRANSACTION does not match
                                        identifier on START OF TRANSACTION.

The following error conditions are transient: 1,2,3,5,6,8; transient errors will eventually go away if the transaction is repeated. The following error conditions are fatal to a subsequent retry of the same transaction: 4,7,9,10,11,14,15,18,19.

INDEX

INDEX

INDEX

INDEX